FFSM++

1.1.0

Generated by Doxygen 1.8.11

Contents

1 FFSM++ Reference Manual (doxygen-generated)

This is the Reference Manual of `FFSM++`.
It contains detailed developer information on the C++ version of the model retrieved automatically from the latest version of the source code (updated daily).
It includes class description, class members, collaboration and caller graphs, as well as the full source code.
Developers can browse the GIT code from its `github web interface`.
Access to git is restricted as it included some input data for which we do not hold copyright and we can't hence redistribute.
If you need access to the source code in a more convenient form (e.g. a zip archive) or to a "cleaned-up" version of the input file please just drop `us` an email.

2 Todo List

Member ModelCore::runBiologicalModule ()

Harvest volumes from death trees

Member ModelCoreSpatial::initializePixelArea ()

here I have finally also area_ft_dc_px and I can implement the new one I am in 2006

: also update area_l

Member ModelData::getScenarioIndex ()

Check that I can call this function all around the model and not only at the beginning

Member ModelRegion::getVolumes ()

Implement me (but really needed?)

Member ModelRegion::getVolumes (int fType_h)

Implement me (but really needed?)

Member ModelRegion::getVolumes (int fType_h, string dClass_h)

Implement me (but really needed?)

Member ThreadManager::run ()

.. perform a better exception handing..

Member UnzipPrivate::extractFile (const QString &path, ZipEntryP &entry, const QDir &dir, UnZip::↩ ExtractionOptions options)

Set creation/last_modified date/time

Member UnzipPrivate::openArchive (QIODevice ∗device)

Ignore CD entry count? CD may be corrupted.

Member ZipPrivate::closeArchive ()

See if we can detect QFile objects using the Qt Meta Object System

SAME AS ABOVE: See if we can detect QFile objects using the Qt Meta Object System

SAME AS ABOVE: See if we can detect QFile objects using the Qt Meta Object System

SAME AS ABOVE: See if we can detect QFile objects using the Qt Meta Object System

Member ZipPrivate::createEntry (const QFileInfo &file, const QString &root, Zip::CompressionLevel level)

Automatic level detection (cpu, extension & file size)

3 Namespace Documentation

3.1 merge_example Namespace Reference

Variables

- list forIFiles
- list prdIFiles
- list carbonIFiles
- list scenarios
- string forOFilename = 'results/forestData_merged.csv'
- string prdOFilename = 'results/productData_merged.csv'
- string carbonOFilename = 'results/carbonBalance_merged.csv'

3.1.1 Variable Documentation

3.1.1.1 list carbonIFiles

Initial value:

```
= [
    ’results/carbonBalance.csv’,
]
```

Definition at line 12 of file merge_example.py.

3.1.1.2 string carbonOFilename = 'results/carbonBalance_merged.csv'

Definition at line 22 of file merge_example.py.

3.1.1.3 list forIFiles

Initial value:

```
= [
    ’results/forestData.csv’,
]
```

Definition at line 6 of file merge_example.py.

3.1.1.4 string forOFilename = 'results/forestData_merged.csv'

Definition at line 20 of file merge_example.py.

3.1.1.5 list prdIFiles

Initial value:

```
= [
    ’results/productData.csv’,
]
```

Definition at line 9 of file merge_example.py.

3.1.1.6 string prdOFilename = 'results/productData_merged.csv'

Definition at line 21 of file merge_example.py.

3.1.1.7 list scenarios

Initial value:

```
= [
  ’test’,
  ’test2’,
]
```

Definition at line 15 of file merge_example.py.

Referenced by ModelData.getScenarioIndex(), ThreadManager.run(), and ThreadManager.runFromConsole().

3.2 merge_lib Namespace Reference

Functions

- def merge (forIFiles_h=[ ], prdIFiles_h=[ ], carbonIFiles_h=[ ], scenarios_h=[ ], forOFilename_h="", prdOFilename_h="", carbonOFilename_h="", variables_h=[ ], regions_h=[ ], years_h=[ ])
- def determinePositions (headerRow)
- def merge_single_file (i_filename_h, o_filename_h, scenarios_h, keepHeader=False, variables_h=[ ], regions_h=[ ], years_h=[ ])

3.2.1 Function Documentation

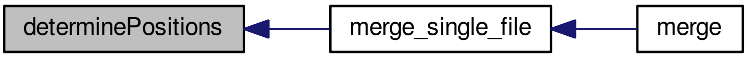
3.2.1.1 def merge_lib.determinePositions (*headerRow*)

Definition at line 29 of file merge_lib.py.

Referenced by merge_single_file().

```
00029 def determinePositions(headerRow):
00030   fields = headerRow.split(’;’)
00031   returnValues = [-1,-1,-1]
00032   for idx, field in enumerate(fields):
00033     if(field == ’parName’ or field == ’balItem’): returnValues[0] = idx
00034     if(field == ’region’):                        returnValues[1] = idx
00035     if(field == ’year’):                          returnValues[2] = idx
00036   if (returnValues[0] == -1 or returnValues[1] == -1 or returnValues[2] == -1):
00037     print ("There has been an error reading the headers of a file.")
00038     exit(1)
00039   return returnValues
00040
00041 # ==============================================================================
```

Here is the caller graph for this function:

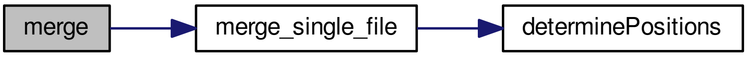


3.2.1.2 def merge_lib.merge (*forIFiles_h* = [], *prdIFiles_h* = [], *carbonIFiles_h* = [], *scenarios_h* = [], *forOFilename_h* = "", *prdOFilename_h* = "", *carbonOFilename_h* = "", *variables_h* = [], *regions_h* = [], *years_h* = [])

Definition at line 5 of file merge_lib.py.

```
00005 def merge(
      forIFiles_h=[],prdIFiles_h=[],carbonIFiles_h=[],scenarios_h=[],forOFilename_h="",prdOFilename_h="",carbonOFilename_h="",
00006   print("*** Processing..")
00007   if len(forIFiles_h)>0:
00008    open(forOFilename_h,'w').close()
00009   if len(prdIFiles_h)>0:
00010     open(prdOFilename_h,'w').close()
00011   if len(carbonIFiles_h)>0:
00012     open(carbonOFilename_h,'w').close()
00013   forCounter=0
00014   prdCounter=0
00015   carbonCounter=0
00016   for forIFile in forIFiles_h:
00017     merge_single_file(forIFile, forOFilename_h, scenarios_h, False if forCounter else True
      , variables_h, regions_h, years_h)
00018     forCounter += 1
00019   for prdIFile in prdIFiles_h:
00020     merge_single_file(prdIFile, prdOFilename_h, scenarios_h, False if prdCounter else True
      , variables_h, regions_h, years_h)
00021     prdCounter += 1
00022   for carbonIFile in carbonIFiles_h:
00023     merge_single_file(carbonIFile, carbonOFilename_h, scenarios_h, False if carbonCounter
      else True, variables_h, regions_h, years_h)
00024     carbonCounter += 1
00025   print ("*** Done!")
00026
00027
00028 # =============================================================================
```

Here is the call graph for this function:



3.2.1.3 def merge_lib.merge_single_file (*i_filename_h*, *o_filename_h*, *scenarios_h*, *keepHeader* = False, *variables_h* = [], *regions_h* = [], *years_h* = [])

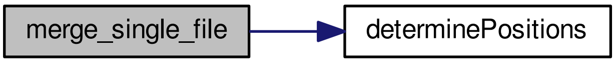
Definition at line 42 of file merge_lib.py.

Referenced by merge().

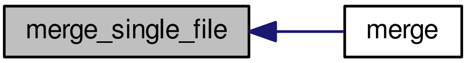
```
00042 def merge_single_file(i_filename_h, o_filename_h, scenarios_h, keepHeader=False,
      variables_h=[], regions_h=[], years_h=[]):
00043   i_file = open(i_filename_h,'r')
00044   o_file = open(o_filename_h,'a')
00045   newRow     =  1
00046   counterRow =  0
00047   parNamePos = -1
00048   regionPos  = -1
00049   yearPos    = -1
00050   positions  = []
00051
00052   while newRow:
00053     row = i_file.readline()
00054     scenarioFilter = False
00055     variableFilter = False
00056     regionFilter   = False
00057     yearFilter     = False
```

```
    finalFilter    = False

    if row == '':
      break
    if(counterRow == 0):
      positions = determinePositions(row)
      parNamePos = positions[0]
      regionPos  = positions[1]
      yearPos    = positions[2]
      if(keepHeader):
        o_file.write(row)
    counterRow += 1
    fields = row.split(';')
    rowScenario = fields[0]

    if(rowScenario in scenarios_h):
      scenarioFilter = True

    if( (len(variables_h) == 0 ) or (fields[parNamePos] in variables_h) ):
      variableFilter = True

    if( (len(regions_h) == 0) or (fields[regionPos] in regions_h) ):
      regionFilter = True

    if( (len(years_h) == 0) or (fields[yearPos] in years_h) ):
      yearFilter = True

    if (scenarioFilter and variableFilter and regionFilter and yearFilter):
      finalFilter = True

    if(finalFilter):
      o_file.write(row)
  i_file.close()
  o_file.close()

```

Here is the call graph for this function:



Here is the caller graph for this function:



3.3 output_parser_example Namespace Reference

Functions

- def main ()
- def override_globals ()
- def printCharts ()
- def printTables ()
- def printAATables ()

3.3.1 Function Documentation

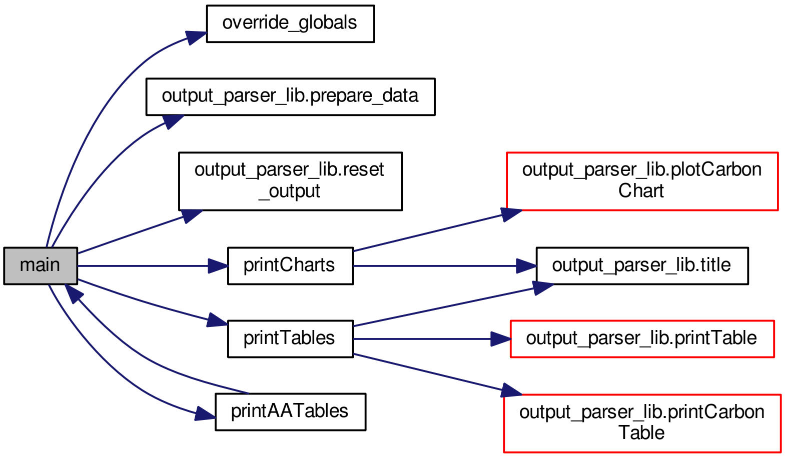
3.3.1.1 def output_parser_example.main ( )

Definition at line 11 of file output_parser_example.py.

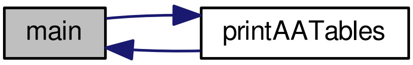
Referenced by printAATables().

```
00011 def main():
00012
00013   override_globals()
00014   prepare_data()
00015   reset_output()
00016
00017   # H - Printing charts
00018   if g.printChartsFlag:
00019     printCharts()
00020
00021   # I - Print tables
00022   if g.printTablesFlag:
00023     printTables()
00024
00025   # L - Print area allocation confrontation
00026   if g.printAATablesFlag:
00027     printAATables()
00028
00029   print "Done!"
00030
00031 # ================================================================================
```

Here is the call graph for this function:



Here is the caller graph for this function:



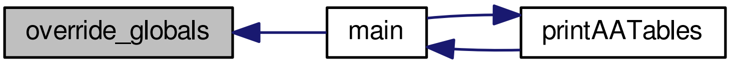
3.3.1.2 def output_parser_example.override_globals ( )

Definition at line 32 of file output_parser_example.py.

Referenced by main().

```
00032 def override_globals():
00033 
00034   g.forIFiles = [
00035     'results/forestData_baseline.csv',
00036     'results/forestData_constant.csv',
00037     'results/forestData_Ph_L.csv',
00038     'results/forestData_Ph_U.csv',
00039     'results/forestData_Pr_C.csv',
00040     'results/forestData_Pr_U.csv',
00041     'results/forestData_Exp_0.csv',
00042     'results/forestData_Exp_1.csv',
00043     'results/forestData_EOL_en_U.csv',
00044   ]
00045 
00046   g.carbonIFiles = [
00047     'results/carbonBalance_baseline.csv',
00048     'results/carbonBalance_constant.csv',
00049     'results/carbonBalance_Ph_L.csv',
00050     'results/carbonBalance_Ph_U.csv',
00051     'results/carbonBalance_Pr_C.csv',
00052     'results/carbonBalance_Pr_U.csv',
00053     'results/carbonBalance_Exp_0.csv',
00054     'results/carbonBalance_Exp_1.csv',
00055     'results/carbonBalance_EOL_en_U.csv',
00056   ]
00057 
00058   g.scenarios = {
00059     'baseline':             '#000000',  # Black
00060     'constant':             '#cccccc',  # Grey
00061     'Ph_L':                 '#b5ff95',  # Light green
00062     'Ph_U':                 '#f40303',  # Red
00063     'Pr_C':                 '#b5ff95',  # Light green
00064     'Pr_U':                 '#f40303',  # Red
00065     'Exp_0':                '#b5ff95',  # Light green
00066     'Exp_1':                '#f40303',  # Red
00067     'EOL_en_U':             '#011bb7',  # Ink blue
00068 
00069   }
00070   g.years = [str(y) for y in range(2007,2101)]     # [2007-2100]
00071   g.printChartsFlag = True
00072   g.printTablesFlag = True
00073   g.printAATablesFlag = False
00074   g.chartoutdir = 'charts'
00075   g.tableoutdir = 'tables'
00076   # key: var short name
00077   # value: turple with long name, unit and optionally variable to act for ponderation and name of
       aggregated variable
00078   g.forVars = {'hV': ['Harvested Volumes', r"$Mm^3$"],
00079        'vReg': ['Regeneration Volumes', r"$Mm^3$"],
00080        'vol': ['Forest Volumes', r"$Mm^3$"],
00081        'expReturns': ['Expected returns','€/ha','forArea', 'totalExpReturns'],
00082        'forArea': ['Forest area','ha'],
00083        'harvestedArea': ['Harvested area','ha'],
00084        'regArea': ['Regeneration area','ha'],
00085        'STOCK_INV': ['Carbon pool in inventoried forest resources', r"$Mt CO_2$"],
00086        'STOCK_EXTRA': ['Carbon pool in non-inventoried forest resources (branches, roots)', r"$Mt CO_2$"],
00087        'STOCK_PRODUCTS': ['Carbon pool in forest products', r"$Mt CO_2$"],
00088        'EM_ENSUB': ['Cumulative emissions from energy substitution', r"$Mt CO_2$"],
00089        'EM_MATSUB': ['Cumulative emissions from material substitution', r"$Mt CO_2$"],
00090        'EM_FOROP': ['Cumulative emissions from forest operations', r"$Mt CO_2$"],
00091   }
00092 
00093 # =============================================================================
```

Here is the caller graph for this function:



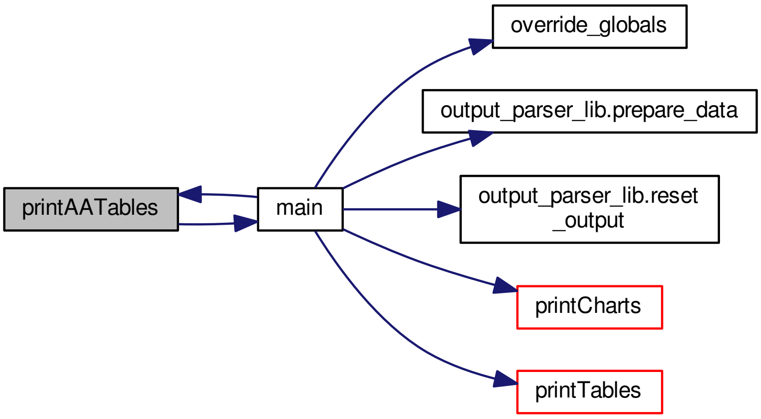
3.3.1.3 def output_parser_example.printAATables ( )

Definition at line 126 of file output_parser_example.py.

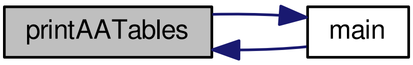
Referenced by main().

```
00126 def printAATables():
00127   print "Printing area allocation tables.."
00128
00129 # ==============================================================================
00130 # EXECUTION ACTUALLY STARTS HERE.....
00131 main()
00132
```

Here is the call graph for this function:


override_globals
output_parser_lib.prepare_data
printAATables
main
output_parser_lib.reset_output
printCharts
printTables

Here is the caller graph for this function:



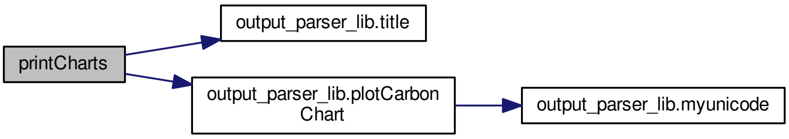
3.3.1.4 def output_parser_example.printCharts ( )

Definition at line 94 of file output_parser_example.py.

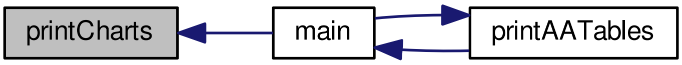
Referenced by main().

```
00094 def printCharts():
00095   print "Printing charts.."
00096
00097   title(’c’,’subsection’, "Carbon charts")
00098   plotCarbonChart([’constant’,’baseline’],’11000’,’’,’cbalance_overall’)
00099   plotCarbonChart([’baseline’,’Exp_0’,’Exp_1’],’11000’,’’,’cbalance_expectations’)
00100   plotCarbonChart([’baseline’,’Pr_C’,’Pr_U’],’11000’,’’,’cbalance_prices’)
00101   plotCarbonChart([’baseline’,’Ph_L’,’Ph_U’],’11000’,’’,’cbalance_ph_impact’)
00102
00103 # ===============================================================================
```

Here is the call graph for this function:


printCharts
output_parser_lib.title
output_parser_lib.plotCarbon Chart
output_parser_lib.myunicode

Here is the caller graph for this function:



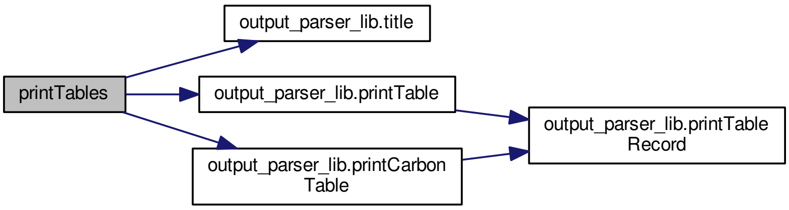
3.3.1.5 def output_parser_example.printTables ( )

Definition at line 104 of file output_parser_example.py.

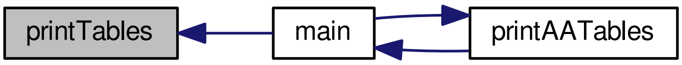
Referenced by main().

```
00104 def printTables():
00105   print "Printing tables.."
00106
00107   y2014_2060 =  [str(y) for y in range(2014,2061)] # [2014-2060]
00108
00109   title(’t’,’section’, "Overall effect")
00110   printTable(’constant’,[’baseline’],[’expReturns’,’vReg’,’vol’,’hV’,’forArea’,’regArea’,’
      harvestedArea’],[’11000’],g.years,’\\texttt{Baseline} vs \\texttt{constant} [avg. 2007-2100]’,’
      cceffect_overall_vars_2007-2100_11000’)
00111   printTable(’constant’,[’baseline’],[’expReturns’,’vReg’,’vol’,’hV’,’forArea’,’regArea’,’
      harvestedArea’],[’11000’],[’2100’],’\\texttt{Baseline} vs \\texttt{constant} [2100]’,’
      cceffect_overall_vars_2100_11000’)
00112   printCarbonTable(’constant’,[’baseline’],’11000’, ’2007’, ’2100’, "\\ce{CO2} balance of
\\texttt{baseline} scenario vs. \\texttt{constant} [yearly avg 2007-2100]",'cceffect_cbalance_2007-2100_11000
', True, True)
  printCarbonTable('constant',['baseline'],'11000', '2013', '2020', "\\ce{CO2} balance of
\\texttt{baseline} scenario vs. \\texttt{constant} [yearly avg 2013-2020]",'cceffect_cbalance_2013-2020_11000
', True, True)

  title('t','section', "Sa on price, physical and expectation effects")
  printTable('baseline',['Pr_C','Pr_U','Ph_L','Ph_U','Exp_0','Exp_1'],['expReturns','vReg','vol',
'hV','forArea','regArea','harvestedArea'],['11000'],g.years,'SA [avg. 2007-2100]','sa_vars_2007-2100_11000',
False)
  printCarbonTable('baseline',['Pr_C','Pr_U','Ph_L','Ph_U','Exp_0','Exp_1'],'11000', '2007'
, '2100', "Sensitivity analisys \\ce{CO2} balance [avg. 2007-2100]",'sa_cbalance_2007-2100_11000', True,
False)
  printTable('baseline',['Pr_C','Pr_U','Ph_L','Ph_U','Exp_0','Exp_1'],['expReturns','vReg','vol',
'hV','forArea','regArea','harvestedArea'],['11000'],y2014_2060,'SA [avg. 2014-2060]','
sa_vars_2014-2060_11000',False)
  printCarbonTable('baseline',['Pr_C','Pr_U','Ph_L','Ph_U','Exp_0','Exp_1'],'11000', '2014'
, '2060', "Sensitivity analisys \\ce{CO2} balance [yearly avg. 2014-2060]",'sa_cbalance_2014-2060_11000',
True, False)
  printTable('baseline',['Pr_C','Pr_U','Ph_L','Ph_U','Exp_0','Exp_1'],['expReturns','vReg','vol',
'hV','forArea','regArea','harvestedArea'],['11000'],['2100'],'SA [2100]','sa_vars_2100_11000',False)

  printCarbonTable('baseline',['EOL_en_U'],'11000', '2007', '2100', "\\ce{CO2} balance of
\\texttt{EOL\\_en\\_U} scenario vs. \\texttt{baseline} [yearly avg 2007-2100]",'
EOL_en_U_cbalance_2007-2100_11000', True, True)



# =============================================================================
```

Here is the call graph for this function:


output_parser_lib.title
printTables
output_parser_lib.printTable
output_parser_lib.printTable Record
output_parser_lib.printCarbon Table

Here is the caller graph for this function:



3.4 output_parser_globals Namespace Reference

Variables

- list forIFiles = [ ]
- list prodIFiles = [ ]
- list carbonIFiles = [ ]

- dictionary scenarios = {}
- list years = [ ]
- bool printChartsFlag = False
- bool printTablesFlag = False
- bool printAATablesFlag = False
- string chartoutdir = 'charts'
- string tableoutdir = 'tables'
- string tablesmaster = '00_master_tables'
- string chartsmaster = '00_master_charts'
- string charttype = 'pdf'
- string sep = ';'
- dictionary countries
- dictionary regions

3.4.1 Variable Documentation

3.4.1.1 list carbonlFiles = [ ]

Definition at line 12 of file output_parser_globals.py.

3.4.1.2 string chartoutdir = 'charts'

Definition at line 20 of file output_parser_globals.py.

3.4.1.3 string chartsmaster = '00_master_charts'

Definition at line 23 of file output_parser_globals.py.

3.4.1.4 string charttype = 'pdf'

Definition at line 24 of file output_parser_globals.py.

3.4.1.5 dictionary countries

Initial value:

```
= {’FRA’: [[’AL (FR42)’, ’AQ (FR61)’, ’AU (FR72)’, ’BN (FR25)’, ’BO (FR26)’, ’BR (FR52)’, ’CE (FR24)’, ’CA (FR21)’,
                ’CO (FR83)’, ’FC (FR43)’, ’HN (FR23)’, ’IF (FR10)’, ’LR (FR81)’, ’LI (FR63)’, ’LO (FR41)’, ’MP (FR62)’,
                ’NP (FR30)’, ’PL (FR51)’, ’PI (FR22)’, ’PC (FR53)’, ’PA (FR82)’, ’RA (FR71)’],’France’]}
```

Definition at line 40 of file output_parser_globals.py.

3.4.1.6 list forlFiles = [ ]

Definition at line 10 of file output_parser_globals.py.

3.4.1.7 bool printAATablesFlag = False

Definition at line 18 of file output_parser_globals.py.

3.4.1.8 bool printChartsFlag = False

Definition at line 16 of file output_parser_globals.py.

3.4.1.9 bool printTablesFlag = False

Definition at line 17 of file output_parser_globals.py.

3.4.1.10 list prodIFiles = [ ]

Definition at line 11 of file output_parser_globals.py.

3.4.1.11 dictionary regions

Initial value:

```
= {'AL (FR42)': 'Alsace', 'AQ (FR61)': 'Aquitaine', 'AU (FR72)': 'Auvergne', 'BN (FR25)': 'Basse-Normandie',
           'BO (FR26)': 'Bourgogne', 'BR (FR52)': 'Bretagne', 'CE (FR24)': 'Centre', 'CA (FR21)': 'Champagne-Ardenne',
           'CO (FR83)': 'Corse', 'FC (FR43)': 'Franche-Comté', 'HN (FR23)': 'Haute-Normandie', 'IF (FR10)': 'Île de France',
           'LR (FR81)': 'Languedoc-Roussillon', 'LI (FR63)': 'Limousin', 'LO (FR41)': 'Lorraine', 'MP (FR62)': 'Midi-Pyrénées',
           'NP (FR30)': 'Nord - Pas-de-Calais', 'PL (FR51)': 'Pays de la Loire', 'PI (FR22)': 'Picardie',
           'PC (FR53)': 'Poitou-Charentes', 'PA (FR82)': 'Provence-Alpes-Côte d\'Azur', 'RA (FR71)': 'Rhône-Alpes'}
```

Definition at line 44 of file output_parser_globals.py.

Referenced by Gis.setSpace().

3.4.1.12 dictionary scenarios = {}

Definition at line 14 of file output_parser_globals.py.

3.4.1.13 string sep = ';'

Definition at line 25 of file output_parser_globals.py.

Referenced by UnzipPrivate.createDirectory().

3.4.1.14 string tableoutdir = 'tables'

Definition at line 21 of file output_parser_globals.py.

3.4.1.15 string tablesmaster = '00_master_tables'

Definition at line 22 of file output_parser_globals.py.

3.4.1.16 list years = [ ]

Definition at line 15 of file output_parser_globals.py.

3.5 output_parser_lib Namespace Reference

Functions

- def prepare_data ()
- def reset_output ()
- def plotMultivariable (scenarios_h, variables_h, region, title, filename, printLegend=True, fwidth=10, fheight=15)
- def plotCarbonChart (scenarios_h, region, title, filename)
- def plotLegend (scenarios_h, filename, title_h="")
- def plotVectorChart_inner (origin, end1, endt, xlabel, ylabel, filename, comp1_color='red', totcomp_color='blue', diffcomp_color='green')
- def printTable (ref_scenario, comparing_scenarios, variables_h, regions_h, years_h, title, filename, singleComparation=False, refYear=0)
- def printAATable (ref_scenarios, comparing_scenarios, regions_h, years_h, title, filename, refYear=0)
- def printCarbonTable (ref_scenario, comparing_scenarios, region, year_start, year_end, title, filename, avg=False, singleComparation=True)
- def printTableRecord (cvar_label, d, el, nscen, valRScenario, valCScenarios, singleComparation)
- def title (cat, level, title)
- def text (cat, text_h)
- def myunicode (astring)

3.5.1 Function Documentation

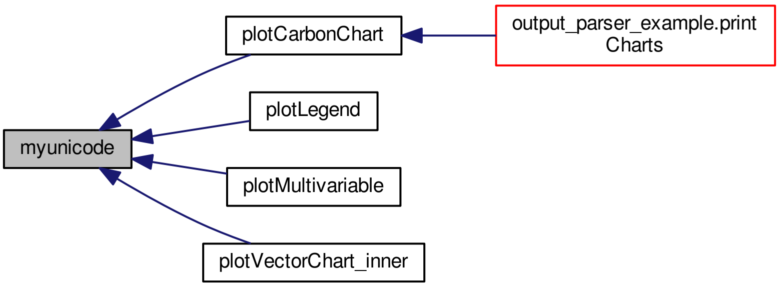
3.5.1.1 def output_parser_lib.myunicode (*astring*)

Definition at line 863 of file output_parser_lib.py.

Referenced by plotCarbonChart(), plotLegend(), plotMultivariable(), and plotVectorChart_inner().

```
00863 def myunicode(astring):
00864   if sys.version_info < (3, 0):
00865     return unicode(astring, 'utf_8')
00866   else:
00867     return astring
00868
```

Here is the caller graph for this function:



3.5.1.2 def output_parser_lib.plotCarbonChart (*scenarios_h, region, title, filename*)

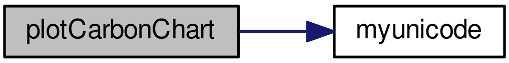
Definition at line 278 of file output_parser_lib.py.

Referenced by output_parser_example.printCharts().

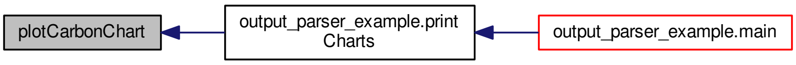
```
00278 def plotCarbonChart(scenarios_h,region,title, filename):
00279 #def plotMultivariable(scenarios_h, variables_h, region, title, filename, printLegend=True):
00280 
00281 
00282   cVariables = [
00283       [’Forest pool’, [’STOCK_INV’,’STOCK_EXTRA’],’:’,3,’#314004’],
00284       [’Wood products pool’, [’STOCK_PRODUCTS’],’--’,3,’#7f0021’],
00285       [’Net cumulative substitution effect’, [’EM_ENSUB’,’EM_MATSUB’,’EM_FOROP’],’-’,4,’#83caff’],
00286   ]
00287 
00288   nscen = len(scenarios_h)
00289 
00290 
00291   matplotlib.rcParams.update({’font.size’: 22})
00292 
00293 
00294   fig = plt.gcf()
00295   fig.set_size_inches(12,10)
00296   ylabel = myunicode("Gt CO2 eq")
00297   plt.title(myunicode(title))
00298   plt.ylabel(ylabel)
00299 
00300   totals = [[0]*len(g.x)]* nscen
00301 
00302   if nscen > 1: #normal line plots
00303     for idg, cGroup in enumerate(cVariables):
00304       for ids, scenario in enumerate(scenarios_h):
00305         grTotals =  [0]*len(g.x)
00306         #serieName = myunicode(cGroup[0] + " - " + scenario)
00307         serieName = "_"+myunicode(scenario) # not shown in legend
00308         if idg==2:
00309           serieName = myunicode(scenario)
00310         serieColor = g.scenarios[scenario]
00311         serieLineType = cGroup[2]
00312         serieWidth = cGroup[3]
00313         for var in cGroup[1]: # for idx, var in enumerate(cGroup[1]):
00314           key = region, var, scenario, ""
00315           varData = g.odata[key]
00316           grTotals = [x2+y for x2, y in zip(grTotals, varData)]
00317 
00318         totals[ids]  = [x3+y2 for x3, y2 in zip(totals[ids],grTotals)]
00319         y = [x4 / 1000 for x4 in totals[ids]]
00320         plt.plot(g.x, y, serieLineType, label=serieName, linewidth=serieWidth, color=serieColor)
00321   else: #area stacked plot
00322     fillColours = []
00323     y = []
00324     for cGroup in cVariables:
00325       y_local =  np.zeros(len(g.x))
00326       fillColour = cGroup[4]
00327       for var in cGroup[1]: # for idx, var in enumerate(cGroup[1]):
00328           key = region, var, scenarios_h[0], ""
00329           varData = np.array(g.odata[key])
00330           #y_local += varData # For some reasons this doesn’t work
00331           y_local = [t+(a/1000) for t, a in zip(y_local, varData)]
00332       y.append(y_local)
00333       fillColours.append(fillColour)
00334     for cGroup in reversed(cVariables):
00335       serieName = myunicode(cGroup[0])
00336       fillColour = cGroup[4]
00337       plt.plot([], [], color=fillColour, linewidth=4, label=serieName) # plotting emply data hack as
       stackplot doesn’t support the legend
00338 
00339     ax = fig.add_subplot(111)
00340     ax.stackplot(g.x, y, colors=fillColours, edgecolor = "none")
00341     ax.autoscale_view(’tight’)
00342 
00343   #plt.legend(loc=’lower right’, ncol=3, shadow=False, labelspacing=0., prop={’size’:12})
00344   plt.legend(loc=’lower right’, ncol=1, shadow=False, labelspacing=0., prop={’size’:14})
00345   #plt.ylim([0,18]) # This would scale the plot y axis to the desired ranges
00346   plt.savefig(g.chartoutdir+"/"+filename+"_"+region+"."+g.charttype, dpi=300)
00347   #plt.show()
00348   plt.close()
00349 
00350   omasterfilename = g.chartoutdir+’/’+g.chartsmaster+’.tex’
00351   omfile = open(omasterfilename,’a’)
00352   omfile.write("\\begin{figure}[htbp]\n")
00353   omfile.write("  \\centering\n")
```

```
00354   omfile.write("  \\caption{"+title+"}\n")
00355   omfile.write("  \\includegraphics[width=0.8\\textwidth]{\""+g.chartoutdir+"/"+filename+"_"+region+"\"}\n"
      )
00356   omfile.write("  \\label{fig:"+filename+"}\n")
00357   omfile.write("\\end{figure}\n")
00358   omfile.close()
00359 
00360 """
00361       scenTotals
00362         y = odata[key]
00363   plt.plot(x, y, serieLineType, label=serieName, linewidth=serieWidth, color=serieColor)
00364     handles, labels = ax.get_legend_handles_labels()
00365   #plt.subplots_adjust(hspace=0.6)
00366   #handles, labels = ax.get_legend_handles_labels()
00367   #ax.legend(handles, labels, ncol=3, shadow=False, title="Legend")
00368   if printLegend:
00369     plt.figlegend(handles, labels, loc = 'lower center', ncol=3, shadow=False, labelspacing=0., prop={'size
      ':12})
00370   #plt.savefig(chartoutdir+"/"+filename+"_"+region+"."+charttype, bbox_inches='tight', dpi=300)
00371   plt.savefig(chartoutdir+"/"+filename+"_"+region+"."+charttype, dpi=300)
00372   #plt.show()
00373   plt.close()
00374 
00375   omasterfilename = chartoutdir+'/'+chartsmaster+'.tex'
00376   omfile = open(omasterfilename,'a')
00377   omfile.write("\\begin{figure}[htbp]\n")
00378   omfile.write("  \\centering\n")
00379   omfile.write("  \\caption{"+title+"}\n")
00380   omfile.write("  \\includegraphics[width=0.8\\textwidth]{"+chartoutdir+"/"+filename+"_"+region+"}\n")
00381   omfile.write("  \\label{fig:"+filename+"}\n")
00382   omfile.write("\\end{figure}\n")
00383   omfile.close()
00384   """
00385 
00386 # ============================================================================
```

Here is the call graph for this function:



Here is the caller graph for this function:


plotCarbonChart
output_parser_example.print Charts
output_parser_example.main

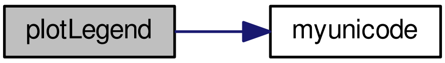
3.5.1.3 def output_parser_lib.plotLegend (*scenarios_h, filename, title_h = ""*)

Definition at line 387 of file output_parser_lib.py.

```
00387 def plotLegend(scenarios_h, filename, title_h=""):
00388   nscen = len(scenarios_h)
00389   fig = plt.gcf()
00390   fheight = (15/15)*nscen+0.2
00391   fig.set_size_inches(10,fheight)
```

```
00392   #ax = plt.axes()
00393   #ax.set_axis_off()
00394
00395   #fig = plt.figure()
00396   ax =fig.add_subplot(111)
00397   ax.set_axis_off()
00398
00399   for spGroup in sorted(g.spAggregates.keys()):
00400     for scenario in scenarios_h:
00401       serieName = myunicode(spGroup + " - " + scenario)
00402       serieColor = g.scenarios[scenario]
00403       serieLineType = g.spAggregates[spGroup][1]
00404       serieWidth = g.spAggregates[spGroup][2]
00405       #print serieName+ " - " + serieLineType + " - " + str(serieWidth)
00406       dummyx = [1]
00407       dummyy = [1]
00408       plt.plot(dummyx, dummyy, serieLineType, label=serieName, linewidth=serieWidth, color=serieColor)
00409   handles, labels = ax.get_legend_handles_labels()
00410   ax.legend(handles, labels, ncol=3, shadow=False) # removed title=title_h
00411   plt.savefig(g.chartoutdir+"/"+filename+"."+g.charttype, bbox_inches='tight', pad_inches=0.1, dpi=300)
00412   #plt.show()
00413   plt.close()
00414
00415   omasterfilename = g.chartoutdir+'/'+g.chartsmaster+'.tex'
00416   omfile = open(omasterfilename,'a')
00417   omfile.write("\\begin{figure}[htbp]\n")
00418   omfile.write("  \\centering\n")
00419   omfile.write("  \\caption{"+title_h+"}\n")
00420   omfile.write("  \\includegraphics[width=0.8\\textwidth]{\""+g.chartoutdir+"/"+filename+"\"}\n")
00421   omfile.write("  \\label{fig:"+filename+"}\n")
00422   omfile.write("\\end{figure}\n")
00423   omfile.close()
00424
00425 #import matplotlib.pyplot as plt
00426 #ax = plt.subplot()  #create the axes
00427 #ax.set_axis_off()  #turn off the axis
00428 #....  #do patches and labels
00429 #ax.legend(patches, labels, ...)  #legend alone in the figure
00430 #plt.show()
00431
00432 # ============================================================================
```

Here is the call graph for this function:



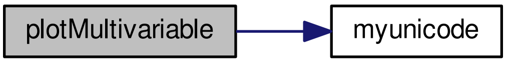
3.5.1.4 def output_parser_lib.plotMultivariable (*scenarios_h*, *variables_h*, *region*, *title*, *filename*, *printLegend* = `True`, *fwidth* = `10`, *fheight* = `15`)

Definition at line 223 of file output_parser_lib.py.

```
00223 def plotMultivariable(scenarios_h, variables_h, region, title, filename, printLegend=True,
       fwidth=10, fheight=15):
00224
00225   nvar = len(variables_h)
00226   nscen = len(scenarios_h)
00227   #plt.figure(1)
00228   fig = plt.gcf()
00229   # suggested: fheight = (15/5)*nvar+0.2
00230   #if nvar == 1:
00231   #  fheight = 4
00232   #if nvar == 2:
00233   #  fheight = 8
00234   fig.set_size_inches(10,fheight) # 15 inches height is fine with 4 variables
00235   maintitle = myunicode(title)
```

```
  handles =[]
  labels = []
  #plt.suptitle(maintitle, fontsize=16, ha='center')
  for i in range(nvar):
    #plt.subplot(nvar,1,i+1)
    ax =fig.add_subplot(nvar,1,i+1)
    subplotTitle = myunicode(g.forVars[variables_h[i]][0])
    ylabel = myunicode(g.forVars[variables_h[i]][1])
    plt.title(subplotTitle)
    plt.ylabel(ylabel)
    for spGroup in sorted(g.spAggregates.keys()):
      for scenario in scenarios_h:
        serieName = myunicode(spGroup + " - " + scenario)
        serieColor = g.scenarios[scenario]
        serieLineType = g.spAggregates[spGroup][1]
        serieWidth = g.spAggregates[spGroup][2]
        #print serieName+ " - " + serieLineType + " - " + str(serieWidth)
        key = region, variables_h[i], scenario, spGroup
        y = g.odata[key]
        plt.plot(g.x, y, serieLineType, label=serieName, linewidth=serieWidth, color=serieColor)
    handles, labels = ax.get_legend_handles_labels()
  #plt.subplots_adjust(hspace=0.6)
  #handles, labels = ax.get_legend_handles_labels()
  #ax.legend(handles, labels, ncol=3, shadow=False, title="Legend")
  if printLegend:
    plt.figlegend(handles, labels, loc = 'lower center', ncol=3, shadow=False, labelspacing=0., prop={'size
':12})
  #plt.savefig(chartoutdir+"/"+filename+"_"+region+"."+charttype, bbox_inches='tight', dpi=300)
  plt.savefig(g.chartoutdir+"/"+filename+"_"+region+"."+g.charttype, dpi=300)
  #plt.show()
  plt.close()

  omasterfilename = g.chartoutdir+'/'+g.chartsmaster+'.tex'
  omfile = open(omasterfilename,'a')
  omfile.write("\\begin{figure}[htbp]\n")
  omfile.write("  \\centering\n")
  omfile.write("  \\caption{"+title+"}\n")
  omfile.write("  \\includegraphics[width=0.8\\textwidth]{\""+g.chartoutdir+"/"+filename+"_"+region+"\"}\n"
)
  omfile.write("  \\label{fig:"+filename+"}\n")
  omfile.write("\\end{figure}\n")
  omfile.close()

# ============================================================================
```

Here is the call graph for this function:



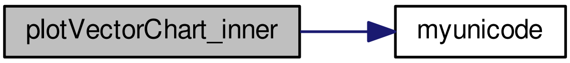
3.5.1.5 def output_parser_lib.plotVectorChart_inner (*origin*, *end1*, *endt*, *xlabel*, *ylabel*, *filename*, *comp1_color* = `'red'`, *totcomp_color* = `'blue'`, *diffcomp_color* = `'green'`)

```
Plot a 2-d vector difference
# @params:
# origin: x and y of the origin of the vectors
# end1:  (x,y) coordinates of the ending of the first component vector
# end2:  (x,y) coordinates of the ending of the total component of the vector
# xlabel: xlabel
# ylabel: ylabel
# filename: filename
# totcomp_color: color (English or #HTML_code) of the vector representing the total component
# comp1_color: color (English or #HTML_code) of the vector representing the first component
# diffcomp_color: color (English or #HTML_code) of the vector representing the difference component
```

Definition at line 433 of file output_parser_lib.py.

```
00433 def plotVectorChart_inner(origin,end1,endt,xlabel,ylabel,filename, comp1_color='red',
      totcomp_color='blue', diffcomp_color='green'):
00434   '''
00435   Plot a 2-d vector difference
00436   # @params:
00437   # origin: x and y of the origin of the vectors
00438   # end1:  (x,y) coordinates of the ending of the first component vector
00439   # end2:  (x,y) coordinates of the ending of the total component of the vector
00440   # xlabel: xlabel
00441   # ylabel: ylabel
00442   # filename: filename
00443   # totcomp_color: color (English or #HTML_code) of the vector representing the total component
00444   # comp1_color: color (English or #HTML_code) of the vector representing the first component
00445   # diffcomp_color: color (English or #HTML_code) of the vector representing the difference component
00446   '''
00447
00448   a   = plt.figure()
00449   ax  = plt.gca()
00450   fig = plt.gcf()
00451   flag_2d = True
00452   if(origin[0] == end1[0] == endt[0]):
00453     flag_2d = False;
00454     fig.set_size_inches(6,10)
00455   else:
00456     fig.set_size_inches(10,10)
00457   end2 = (endt[0]-end1[0]+origin[0],endt[1]-end1[1]+origin[1])
00458   minx = min(origin[0],end1[0],end2[0],endt[0])
00459   maxx = max(origin[0],end1[0],end2[0],endt[0])
00460   miny = min(origin[1],end1[1],end2[1],endt[1])
00461   maxy = max(origin[1],end1[1],end2[1],endt[1])
00462   centre = (((maxx-minx)/2)+minx,((maxy-miny)/2)+miny)
00463
00464   # This allows to write a serie of arrows in one go, but didn't got how in this case colours work
00465   #X  = (origin[0], origin[0], origin[0])
00466   #Y  = (origin[1], origin[1], origin[1])
00467   #X2 = (end1[0]-origin[0], endt[0]-origin[0], end2[0]-origin[0])
00468   #Y2 = (end1[1]-origin[1], endt[1]-origin[1], end2[1]-origin[1])
00469   #C  = (255,10,150) # ? colour codes, but didn't got it
00470   # ax.quiver(X,Y,X2,Y2,Cangles='xy',scale_units='xy',scale=1, width=0.008)
00471
00472   # Printing first component..
00473   ax.quiver(origin[0],origin[1],end1[0]-origin[0],end1[1]-origin[1],angles='xy',scale_units='xy',scale=1,
      width=0.008, color=comp1_color)
00474   # Printing total component..
00475   ax.quiver(origin[0],origin[1],endt[0]-origin[0],endt[1]-origin[1],angles='xy',scale_units='xy',scale=1,
      width=0.008, color=totcomp_color)
00476   # Printing diff component..
00477   ax.quiver(origin[0],origin[1],end2[0]-origin[0],end2[1]-origin[1],angles='xy',scale_units='xy',scale=1,
      width=0.008, color=diffcomp_color)
00478
00479   x  = (end1[0],end2[0])
00480   y  = (end1[1],end2[1])
00481   x2 = (endt[0]-end1[0], endt[0]-end2[0])
00482   y2 = (endt[1]-end1[1], endt[1]-end2[1])
00483
00484   if(flag_2d):
00485     ax.quiver(x,y,x2,y2,angles='xy',scale_units='xy',scale=1, width=0.005, color='gray')
00486     ax.set_xlim([minx- (centre[0]-minx)*0.4, maxx + (maxx-centre[0])*0.4])
00487
00488   ax.set_ylim([miny- (centre[1]-miny)*0.4, maxy + (maxy-centre[1])*0.4])
00489
00490   plt.xlabel(myunicode(xlabel))
00491   plt.ylabel(myunicode(ylabel))
00492   # Uncomment the following lines if you want to display instead of save the figure..
00493   #plt.draw()
00494   #plt.show()
00495   plt.savefig(filename, dpi=300, transparent=False, bbox_inches='tight', pad_inches=0.1)
00496
00497 # =============================================================================
```

Here is the call graph for this function:



3.5.1.6 def output_parser_lib.prepare_data ( )

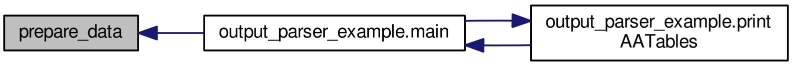
Definition at line 19 of file output_parser_lib.py.

Referenced by output_parser_example.main().

```
00019 def prepare_data():
00020  #print ("Loading and preparing the data..")
00021
00022  # A - creating empty dictionaries with just the keys..
00023  for country, data in g.countries.items():
00024   g.regions[country] = data[1] # add 11000: 'France' to regions
00025  g.sortedregions = sorted(g.regions)
00026  #k = d.keys(); k.sort(). Use k = sorted(d)
00027
00028  specieswithAggregates = g.spGroups
00029  specieswithAggregates.extend(g.spAggregates.keys())
00030  tempSpecieswithAggregates = specieswithAggregates
00031  #tempSpecieswithAggregates.append("") # attenction that python doesn not create a new variable, just
      alias the two
00032  tempSpGroups = g.spGroups
00033  tempSpGroups.append("")
00034
00035
00036  variablesWithAggregates = list(g.forVars.keys())
00037  for variable in g.forVars.keys():
00038   #'expReturns': ['Expected returns','€/ha','forArea','totalExpReturns','globalft'],
00039   if len(g.forVars[variable]) >= 3:
00040    variablesWithAggregates.append(g.forVars[variable][3])
00041
00042  for region in g.regions.keys():
00043   for variable in variablesWithAggregates:
00044    for scenario in g.scenarios.keys():
00045     for spGroup in tempSpecieswithAggregates:
00046      for year in g.years:
00047       key = region, variable, scenario, spGroup, year
00048       g.idata[key] = 0.0
00049  for region in g.regions.keys():
00050   for variable in variablesWithAggregates:
00051    for scenario in g.scenarios.keys():
00052     for spGroup in tempSpecieswithAggregates:
00053      key = region, variable, scenario, spGroup
00054      g.odata[key] = []
00055  for year in g.years:
00056   g.x.append(int(year))
00057
00058
00059  # B - loading data..
00060  for ifile in g.forIFiles:
00061   idata_raw = csv.DictReader(open(ifile, 'r'), delimiter=g.sep)
00062   for rec in idata_raw:
00063    # scen;parName;country;region;forType;diamClass;year;value;
00064    iForType  = rec['forType']
00065    if iForType == 'broadL':
00066     debug = True
00067    for spAggregateKey, spAggregate in g.spAggregates.items():
00068     if (len(spAggregate) >= 3 and iForType ==  spAggregate[3]):
00069      iForType = spAggregateKey
00070      break
00071    key = rec['region'],rec['parName'],rec['scen'],iForType,rec['year']
00072    if key in g.idata:
00073     g.idata[key] += float (rec['value'])
00074  debug = g.idata
```

```
  for ifile in g.prodIFiles:
    idata_raw = csv.DictReader(open(ifile, ’r’), delimiter=g.sep)
    for rec in idata_raw:
      # scen;parName;country;region;prod;freeDim;year;value;
      key = rec[’region’],rec[’parName’],rec[’scen’],rec[’prod’],rec[’year’]
      if key in g.idata:
        g.idata[key] += float (rec[’value’])

  for ifile in g.carbonIFiles:
    #print (g.carbonIFiles)
    idata_raw = csv.DictReader(open(ifile, ’r’), delimiter=g.sep)
    for rec in idata_raw:
      # scen;parName;country;region;forType;diamClass;year;value;
      key = rec[’region’],rec[’balItem’],rec[’scen’],"",rec[’year’]
      #print key
      if key in g.idata:
        g.idata[key] += float (rec[’value’])
        #print (key)
        #print (g.idata[key])

  #exit(1)

  # C - creating aggregated data for variables that need to be pondered
#  for variable in g.forVars.keys():
#    #’expReturns’: [’Expected returns’,’€/ha’,’forArea’, ’totalExpReturns’],
#    if len(g.forVars[variable]) >= 3:
#      pondVariable = g.forVars[variable][2]
#      totalVariable = g.forVars[variable][3]
#      for region in g.regions.keys():
#        for scenario in g.scenarios.keys():
#          for spGroup in specieswithAggregates:
#            for year in g.years:
#              key = region, variable, scenario, spGroup, year
#              key_tvar = region, totalVariable, scenario, spGroup, year
#              if(g.forVars[variable][4] == ’sameft’):
#                  key_pvar = region, pondVariable, scenario, spGroup, year
#                  g.idata[key_tvar] = g.idata[key] * g.idata[key_pvar]
#              elif(g.forVars[variable][4] == ’globalft’):
#                  totalPvar = 0.0;
#                  for spGroup2 in g.spGroups:
#                      key_pvar = region, pondVariable, scenario, spGroup2, year
#                      totalPvar +=g.idata[key_pvar]
#                  g.idata[key_tvar] = g.idata[key] * totalPvar
#              else:
#                  print("Error, I don’t know how to handle this ponderation method: "+g.forVars[variable][4])
#                  exit(1)

  # D - performing various summing up..

  # summing up the specie aggregation
  for spAggregate, species in g.spAggregates.items():
    for region in g.regions.keys():
      for variable in variablesWithAggregates:
        if(variable != ’expReturns’ and variable != ’sumExpReturns’): # let’s skip these as the sumExpReturns at group/forest levels are already exogenously read as these are not the sums
         for scenario in g.scenarios.keys():
           for year in g.years:
             destKey = region, variable, scenario, spAggregate, year
             g.idata[destKey] = 0.0
             for specie in species[0]:
               varToBeSumKey = region, variable, scenario, specie, year
               g.idata[destKey] +=  g.idata[varToBeSumKey]

  # summing up to the country level..
  for country, regionsInTheCountry in g.countries.items():
    for variable in variablesWithAggregates:
      for scenario in g.scenarios.keys():
        for spGroup in tempSpGroups:
          for year in g.years:
            destKey = country, variable, scenario, spGroup, year
            g.idata[destKey] = 0.0
            for regionInTheCountry in regionsInTheCountry[0]:
              varToBeSumKey = regionInTheCountry, variable, scenario, spGroup, year
              g.idata[destKey] +=  g.idata[varToBeSumKey]

    # Correcting the country aggregation of expected returns
    for scenario in g.scenarios.keys():
      for spGroup in tempSpGroups:
        for year in g.years:
          countryForArea_key = country,’forArea’,scenario,’00_Total’,year
          countrySumExpReturns_key = country, ’sumExpReturns’, scenario, spGroup, year
          target_key = country,’expReturns’, scenario, spGroup, year
          g.idata[target_key] = g.idata[countrySumExpReturns_key]/ g.idata[countryForArea_key]

  # checking country aggregation, ok
  #for country, regionsInTheCountry in countries.iteritems():
    #print "country: " + country + " " + str(idata[country,'vol', 'vRegFixed', 'broadL_highF', '2006'])
    #for regionInTheCountry in regionsInTheCountry[0]:
      #print "region: " + regionInTheCountry + " " + str(idata[regionInTheCountry,'vol', 'vRegFixed',
 'broadL_highF', '2006'])



  # testing specie aggregating
  #for spAggregate, species in spAggregates.iteritems():
    #print "aggregate: "+ spAggregate + " " + str(idata['11042','vol', 'vRegFixed', spAggregate, '2006'])
    #for specie in species[0]:
      #print "specieGroup: " + specie + " " + str(idata['11042','vol', 'vRegFixed', specie, '2006'])

#  # E - after all the summing up let's compute the poundered value
#  for variable in g.forVars.keys():
#    #'expReturns': ['Expected returns','€/ha','forArea', 'totalExpReturns'],
#    if len(g.forVars[variable]) >= 3:
#      pondVariable = g.forVars[variable][2]
#      totalVariable = g.forVars[variable][3]
#      for region in g.regions.keys():
#        for scenario in g.scenarios.keys():
#          for spGroup in specieswithAggregates:
#            for year in g.years:
#              key = region, variable, scenario, spGroup, year
#              key_pvar = region, pondVariable, scenario, spGroup, year
#              key_tvar = region, totalVariable, scenario, spGroup, year
#              g.idata[key] =  (g.idata[key_tvar] / g.idata[key_pvar]) if  g.idata[key_pvar] != 0 else 0

  # testing ponderation variables
  #for variable in variables.keys():
    #'expReturns': ['Expected returns','€/ha','forArea', 'totalExpReturns'],
    #if len(variables[variable]) >= 3:
      #pondVariable = variables[variable][2]
      #totalVariable = variables[variable][3]
      #print "Orig variable: " + variable + " " + str(idata['11000', variable, 'vRegFixed','Total',
 '2006'])
      #print "Pond variable: " + pondVariable + " " + str(idata['11000', pondVariable, 'vRegFixed',
 'Total', '2006'])
      #print "Total variable: "  + totalVariable + " " + str(idata['11000', totalVariable, 'vRegFixed',
 'Total', '2006'])

  # F - converting everything in years array
  for region in g.regions.keys():
    for variable in variablesWithAggregates:
      for scenario in g.scenarios.keys():
        for spGroup in tempSpecieswithAggregates:
          key = region, variable, scenario, spGroup
          for year in g.years:
            key_year = region, variable, scenario, spGroup, year
            g.odata[key].append(g.idata[key_year])

  # testing odata
  #print "idata[2005]: " + str(idata['11000', 'vol', 'vRegFixed','Total', '2005'])
  #print "idata[2006]: " + str(idata['11000', 'vol', 'vRegFixed','Total', '2006'])
  #print "odata: " + str(odata['11000', 'vol', 'vRegFixed','Total'])

# =============================================================================
```

Here is the caller graph for this function:


prepare_data
output_parser_example.main
output_parser_example.print AATables

3.5.1.7 def output_parser_lib.printAATable (*ref_scenarios*, *comparing_scenarios*, *regions_h*, *years_h*, *title*, *filename*, *refYear* = 0)

Definition at line 603 of file output_parser_lib.py.

```
def printAATable(ref_scenarios, comparing_scenarios, regions_h, years_h, title, filename, refYear=0) :
#def printTable(ref_scenario, comparing_scenarios, variables_h, regions_h, years_h, title, filename):

#printAATable(['cc1','cc1_nospvar','cc2','cc2_nospvar','cc3','cc3_nospvar','cc3','cc3_nospvar'],['bau','bau_nospvar','ba
 allocation [% variation over bau]', 'area_allocation')
  d = " & "
  el = " \\\\"

  scenario_labels = []
  nscen = len(ref_scenarios)
  nscen_comp = len(comparing_scenarios)
  if nscen != nscen_comp:
    print ("Error in printAATable: number of comparing vs reference scenarios must be the same !")
    exit(1)
  nyears = len(years_h)
  nregions = len(regions_h)
  ntotcol = nscen+1
  for scenario in comparing_scenarios:
    scenario_labels.append(scenario.replace("_", "\\_"))


  oString = ""
  oString += "\\begin{table}[htbp]\n"
  oString += "\\begin{center}\n"
  oString += "\\begin{threeparttable}\n"
  oString += "\\centering\n"
  oString += "\\caption{"+title.replace("_", "\\_").replace("%", "\\%")+"}\n"
  oString += "\\begin{footnotesize}\n"
  oString += "\\begin{tabularx}{\\textwidth}{l "
  for i in range(nscen):
    oString += " r"
  oString += "}\n"

  oString += "\\hline\n"
  oString += "Region"
  for scenario in scenario_labels:
    oString += d+scenario
  oString += el+'\n'
  for spGroup in sorted(g.spAggregates.keys()):
    oString += "\\multicolumn{"+str(ntotcol)+"}{l}{"+spGroup.replace("_", "\\_")+"}"+el+'\n'
    for region in regions_h:
      oString += g.regions[region]
      for s in range(len(comparing_scenarios)):
        sum_value_b = 0.0
        sum_value_c = 0.0
        for year in years_h:
          rYear = str(refYear) if refYear else year # If we overrided the reference year we gonna pick it up here
          key_b = region, 'forArea', ref_scenarios[s], spGroup, rYear
          key_c = region, 'forArea', comparing_scenarios[s], spGroup, year
          sum_value_b += g.idata[key_b]
          sum_value_c += g.idata[key_c]
        reldiff = (100*(sum_value_c-sum_value_b)/sum_value_b) if  sum_value_b != 0 else 0
        oString +=  d+"%+0.3f"%(reldiff)
      oString += el+'\n'


  oString += "\\hline\n"
  oString += "\\end{tabularx}\n"
  oString += "\\end{footnotesize}\n"
  oString += "\\label{tab:"+filename+"}\n"
  oString += "\\end{threeparttable}\n"
  oString += "\\end{center}\n"
  oString += "\\end{table}\n"

  ofilename = g.tableoutdir+'/'+filename+'.tex'
  ofile = open(ofilename,'w')
  ofile.write(oString)
  ofile.close()

  omasterfilename = g.tableoutdir+'/'+g.tablesmaster+'.tex'
  omfile = open(omasterfilename,'a')
  omfile.write("\\input{\""+g.tableoutdir+'/'+filename+".tex\"}\n")
  omfile.close()

# ==============================================================================
```

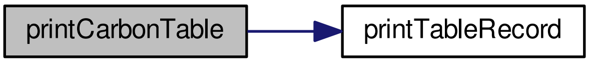
3.5.1.8 def output_parser_lib.printCarbonTable (*ref_scenario*, *comparing_scenarios*, *region*, *year_start*, *year_end*, *title*, *filename*, *avg* = `False`, *singleComparation* = `True`)

Definition at line 676 of file output_parser_lib.py.

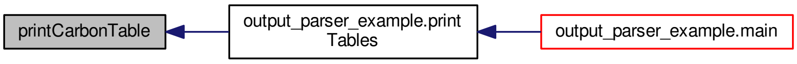
Referenced by output_parser_example.printTables().

```
def printCarbonTable(ref_scenario, comparing_scenarios, region, year_start, year_end,
title, filename, avg=False, singleComparation=True ) :
  #Print carbon balance
  # @params:
  # avg:               true  => output is the yearly average in the period,
  #                    false => output is the difference between year_start and year_end
  # singleComparation: true  => comparing scenarios are seens as repetition of a unique scenario, hence
 stats on their variance is performed,
  #                    false => each comparing scenarios is presented indipendently
  d = " & "
  el = " \\\\"

  cvariables = [
      ['Pools', "- Total pools", [
         ['STOCK_INV',"- Inventoried forest pool"],
         ['STOCK_EXTRA',"- Extra forest pool (branches and roots)"],
         ['STOCK_PRODUCTS',"- Wood products pool"]
         ]],
      ['Emissions', "- Net substitution",
        [['EM_ENSUB',"- Energy substitution"],
         ['EM_MATSUB',"- Material substitution"],
         ['EM_FOROP',"- Emissions from forest operations"]
         ]],
  ]

  label_comparing_scenario = "comparing scenarios"
  labels_comparing_scenarios = []
  nscen = len(comparing_scenarios)
  nyears = (int(year_end) - int(year_start) + 1) if avg else 1
  ncol = 4
  label_ref_scenario = ref_scenario.replace("_", "\\_")

  for comp_scenario in comparing_scenarios:
    labels_comparing_scenarios.append(comp_scenario.replace("_", "\\_"))

  if (singleComparation and nscen == 1):
    label_comparing_scenario = labels_comparing_scenarios[0]

  if (singleComparation):
    if nscen > 2:
      ncol = 5
  else:
    ncol = nscen+2

  oString = ""
  oString += "\\begin{table*}[!htbp]\n"
  oString += "\\begin{center}\n"
  oString += "\\begin{threeparttable}\n"
  oString += "\\centering\n"
  oString += "\\caption{"+title+"}\n"
  oString += "\\begin{footnotesize}\n"
  oString += "\\begin{tabularx}{\\textwidth}{l "
  for nc in range(1,ncol):
    oString += "r "
  oString += "}\n"
  oString += "\\hline\n"

  if (singleComparation):
    if nscen > 2:
      oString += d+"\\texttt{"+label_ref_scenario+"}"+d+"\\texttt{"+label_comparing_scenario+"}"+d+"
difference"+d+"cv"+el+"\n"
    else:
      oString += d+"\\texttt{"+label_ref_scenario+"}"+d+"\\texttt{"+label_comparing_scenario+"}"+d+"
difference"+el+"\n"
  else:
    oString += d+label_ref_scenario
    for label_comparing_scenarios in labels_comparing_scenarios:
      oString += d+label_comparing_scenarios
    oString += el+'\n'

  if(nyears > 1):
    oString += "\\multicolumn{"+str(ncol)+"}{l}{Carbon balance ($Mt~ \ce{CO2}eq.~y^{-1}$)}"+el+'\n'
  else:
    oString += "\\multicolumn{"+str(ncol)+"}{l}{Carbon balance ($Mt~ \ce{CO2}eq.)$}"+el+'\n'

  # Total totals..
  totSumValRScenario = 0
  totSumValCScenarios = [0] * nscen
  for vargroup in cvariables:
    # Group totals..
    grSumValRScenario = 0
    grSumValCScenarios = [0] * nscen
    oString += "\\multicolumn{"+str(ncol)+"}{l}{"+vargroup[0]+"}"+el+'\n'
    # Working on the single variables..
    for cvar in vargroup[2]:
      cvar_name          = cvar[0]
      cvar_label         = cvar[1]
      valRScenario        = (g.idata[region, cvar_name, ref_scenario, "", year_end]-g.idata[region, cvar_name, ref_scenario, "", year_start])/nyears
      grSumValRScenario  += valRScenario
      totSumValRScenario += valRScenario
      valCScenarios       = [0] * nscen

      for s in range(nscen):
        valCScenarios[s] = (g.idata[region, cvar_name, comparing_scenarios[s], "", year_end]-g.idata[region, cvar_name, comparing_scenarios[s], "", year_start])/nyears
        grSumValCScenarios[s] += valCScenarios[s]
        totSumValCScenarios[s] += valCScenarios[s]
      oString += printTableRecord(cvar_label, d, el, nscen, valRScenario, valCScenarios, singleComparation)
    oString += printTableRecord(vargroup[1], d, el, nscen, grSumValRScenario, grSumValCScenarios,singleComparation)
  oString += printTableRecord("Total \ce{CO2} balance", d, el, nscen, totSumValRScenario, totSumValCScenarios,singleComparation)

  oString += "\\hline\n"
  oString += "\\end{tabularx}\n"
  oString += "\\end{footnotesize}\n"
  oString += "\\label{tab:"+filename+"}\n"
  if (singleComparation and nscen > 2):
    oString += "\\begin{tablenotes}\n"
    oString += "\\begin{footnotesize}\n"
    oString += "\\item [a] Significantly different from 0 at $\\alpha=0.01$\n"
    oString += "\\item [b] Significantly different from 0 at $\\alpha=0.001$\n"
    oString += "\\end{footnotesize}\n"
    oString += "\\end{tablenotes}\n"
  oString += "\\end{threeparttable}\n"
  oString += "\\end{center}\n"
  oString += "\\end{table*}\n"

  ofilename = g.tableoutdir+’/’+filename+’.tex’
  ofile = open(ofilename,’w’)
  ofile.write(oString)
  ofile.close()

  omasterfilename = g.tableoutdir+’/’+g.tablesmaster+’.tex’
  omfile = open(omasterfilename,’a’)
  omfile.write("\\input{\""+g.tableoutdir+’/’+filename+".tex\"}\n")
  omfile.close()
# =============================================================================
```

Here is the call graph for this function:



Here is the caller graph for this function:


printCarbonTable
output_parser_example.print Tables
output_parser_example.main

3.5.1.9 def output_parser_lib.printTable (*ref_scenario*, *comparing_scenarios*, *variables_h*, *regions_h*, *years_h*, *title*, *filename*, *singleComparation* = `False`, *refYear* = `0`)

`Print a LaTeX Table for variables variable_h comparing ref_scenario scenario vs coparing_scenarios.`

```
@param singleComparation: if True multiple comparing scenarios are treated as multiple replications of the sam
some basic stats are computed; if False they are all represented as diff from the ref_scenario.
@param refYear: if 0 reference vs comparing scenarios are compared on the same year (or average of years if ye
Otherwise the comparing scneario at year(s) years_h is compared with reference scenario at year refYear (usefu
effects within a single scenario)
```

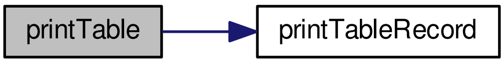
Definition at line 498 of file output_parser_lib.py.

Referenced by output_parser_example.printTables().

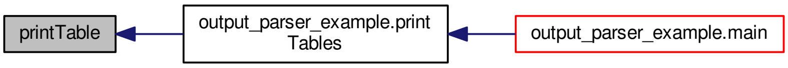
```
00498 def printTable(ref_scenario, comparing_scenarios, variables_h, regions_h, years_h, title,
      filename, singleComparation=False, refYear=0):
00499   """Print a LaTeX Table for variables variable_h comparing ref_scenario scenario vs coparing_scenarios.
00500   @param singleComparation: if True multiple comparing scenarios are treated as multiple replications of
       the same scenario and
00501   some basic stats are computed; if False they are all represented as diff from the ref_scenario.
00502   @param refYear: if 0 reference vs comparing scenarios are compared on the same year (or average of years
       if years_h has length > 1.).
00503   Otherwise the comparing scneario at year(s) years_h is compared with reference scenario at year refYear
       (useful to see the dynamic
00504   effects within a single scenario)
00505   """
00506   d = " & "
00507   el = " \\\\"
00508   label_comparing_scenario = "comparing scenarios"
00509   labels_comparing_scenarios = []
00510   nvar = len(variables_h)
00511   nscen = len(comparing_scenarios)
00512   nyears = len(years_h)
00513   nregions = len(regions_h)
00514   ncol = 4
00515   label_ref_scenario = ref_scenario.replace("_", "\\_")
00516
00517   for comp_scenario in comparing_scenarios:
00518     labels_comparing_scenarios.append(comp_scenario.replace("_", "\\_"))
00519
00520   if (singleComparation and nscen == 1):
00521     label_comparing_scenario = labels_comparing_scenarios[0]
00522
00523   if (singleComparation):
00524     if nscen > 2:
00525       ncol = 5
00526   else:
00527     ncol = nscen+2 #+1 for the val label and +1 for the ref scenario
00528
00529   oString = ""
00530   oString += "\\begin{table}[htbp]\n"
00531   oString += "\\begin{center}\n"
00532   oString += "\\begin{threeparttable}\n"
00533   oString += "\\centering\n"
00534   oString += "\\caption{"+title+"}\n"
00535   oString += "\\begin{footnotesize}\n"
00536   oString += "\\begin{tabularx}{\\textwidth}{l "
00537   for nc in range(1,ncol):
00538     oString += "r "
00539   oString += "}\n"
00540   oString += "\\hline\n"
00541   if (singleComparation):
00542     if nscen > 2:
00543       oString += d+label_ref_scenario+d+label_comparing_scenario+d+"difference"+d+"cv"+el+"\n"
00544     else:
00545       oString += d+label_ref_scenario+d+label_comparing_scenario+d+"difference"+el+"\n"
00546   else:
00547     oString += d+label_ref_scenario
00548     for label_comparing_scenarios in labels_comparing_scenarios:
00549       oString += d+label_comparing_scenarios
00550     oString += el+'\n'
00551
00552   for region in regions_h:
00553     oString += "\\hline\n"
00554     if nregions > 1:
00555       oString += "\\multicolumn{"+str(ncol)+"}{l}{"+regions[region]+"}"+el+'\n'
00556
00557     for variable in variables_h:
00558       oString += "\\multicolumn{"+str(ncol)+"}{l}{"+g.forVars[variable][0]+" (\\textit{"+g.forVars[variable
      ][1]+"})}"+el+'\n'
00559       for spGroup in sorted(g.spAggregates.keys()):
00560         outSpGroup = spGroup.replace("_", "\\_")
00561         sumRScenario =   0
00562         sumCScenarios = [0] * nscen
00563         valRScenario = 0
00564         valCScenarios = [0] * nscen
```

```
        for year in years_h:
          rYear = str(refYear) if refYear else year # If we overrided the reference year we gonna pick it up here
          keyr = region, variable, ref_scenario, spGroup, rYear
          sumRScenario += g.idata[keyr]
          for s in range(nscen):
            keyc = region, variable, comparing_scenarios[s], spGroup, year
            sumCScenarios[s] += g.idata[keyc]
        valRScenario = sumRScenario/nyears
        for s in range(nscen):
          valCScenarios[s] = sumCScenarios[s]/nyears
        oString += printTableRecord("- "+outSpGroup, d, el, nscen, valRScenario, valCScenarios, singleComparation)

  oString += "\\hline\n"
  oString += "\\end{tabularx}\n"
  oString += "\\end{footnotesize}\n"
  oString += "\\label{tab:"+filename+"}\n"
  if (singleComparation and nscen > 2):
    oString += "\\begin{tablenotes}\n"
    oString += "\\begin{footnotesize}\n"
    oString += "\\item [a] Significantly different from 0 at $\\alpha=0.01$\n"
    oString += "\\item [b] Significantly different from 0 at $\\alpha=0.001$\n"
    oString += "\\end{footnotesize}\n"
    oString += "\\end{tablenotes}\n"
  oString += "\\end{threeparttable}\n"
  oString += "\\end{center}\n"
  oString += "\\end{table}\n"

  ofilename = g.tableoutdir+'/'+filename+'.tex'
  ofile = open(ofilename,'w')
  ofile.write(oString)
  ofile.close()

  omasterfilename = g.tableoutdir+'/'+g.tablesmaster+'.tex'
  omfile = open(omasterfilename,'a')
  omfile.write("\\input{\""+g.tableoutdir+'/'+filename+".tex\"}\n")
  omfile.close()

# =============================================================================
```

Here is the call graph for this function:



Here is the caller graph for this function:


printTable
output_parser_example.print Tables
output_parser_example.main

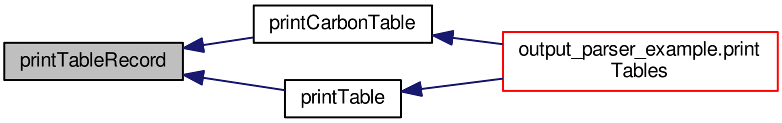
3.5.1.10 def output_parser_lib.printTableRecord (*cvar_label, d, el, nscen, valRScenario, valCScenarios, singleComparation*)

Definition at line 797 of file output_parser_lib.py.

Referenced by printCarbonTable(), and printTable().

```
00797 def printTableRecord(cvar_label, d, el, nscen, valRScenario, valCScenarios,
      singleComparation):
00798
00799   oString = ""
00800   if singleComparation:
00801     avgCScenarios = sum(valCScenarios) / float(nscen)
00802     scenarioDiff = avgCScenarios-valRScenario
00803     scenarioRelativeDiff = 100 * scenarioDiff/valRScenario if valRScenario else 0.0
00804     if nscen > 2:
00805       significance = ""
00806       qdiffCScenarios = [0] * nscen
00807       sumqdiffCScenarios = 0
00808       for s in range(nscen):
00809         qdiffCScenarios[s] = (valCScenarios[s] - avgCScenarios)**2.0
00810         sumqdiffCScenarios += qdiffCScenarios[s]
00811       sd = (sumqdiffCScenarios/(nscen-1))**0.5
00812       t = abs(scenarioDiff)*nscen**0.5/sd if sd>0.0 else 0.0
00813       cv = 100.0 * sd/abs(avgCScenarios) if abs(avgCScenarios)> 0.0 else 0.0
00814       if t >= g.tvalue001[nscen-1-1]:
00815         significance = '$^a$'
00816       if t >= g.tvalue0001[nscen-1-1]:
00817         significance = '$^b$'
00818       oString += cvar_label+d+"%0.3f"%(valRScenario)+d+"%0.3f"%(avgCScenarios)+d+"%0.3f"%(scenarioDiff)+
      significance+' ('+"%0.3f"%(scenarioRelativeDiff)+'\\%)'+d+"%0.2f"%(cv)+' \\%'+el+'\n'
00819     else:
00820       oString += cvar_label+d+"%0.3f"%(valRScenario)+d+"%0.3f"%(avgCScenarios)+d+"%0.3f"%(scenarioDiff)+' (
      '+"%0.2f"%(scenarioRelativeDiff)+'\\%)'+el+'\n'
00821   else:
00822     oString += cvar_label+d+"%0.3f"%(valRScenario)
00823     for valCScenario in valCScenarios:
00824       scenarioDiff = valCScenario-valRScenario
00825       scenarioRelativeDiff = 100 * scenarioDiff/valRScenario if valRScenario else 0.0
00826       oString += d+"%0.2f"%(scenarioRelativeDiff)+'\\%'
00827     oString += el + '\n'
00828   return oString
00829
00830
00831
00832 # ============================================================================
```

Here is the caller graph for this function:



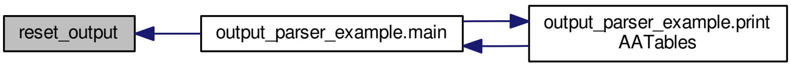
3.5.1.11 def output_parser_lib.reset_output ( )

Definition at line 213 of file output_parser_lib.py.

Referenced by output_parser_example.main().

```
00213 def reset_output():
00214   # G - Reset latex files
00215   filename_t = g.tableoutdir+'/'+g.tablesmaster+'.tex'
00216   filename_c = g.chartoutdir+'/'+g.chartsmaster+'.tex'
00217   file_t = open(filename_t,'w')
00218   file_c = open(filename_c,'w')
00219   file_t.close()
00220   file_c.close()
00221
00222 # ============================================================================
```

Here is the caller graph for this function:


reset_output
output_parser_example.main
output_parser_example.print AATables

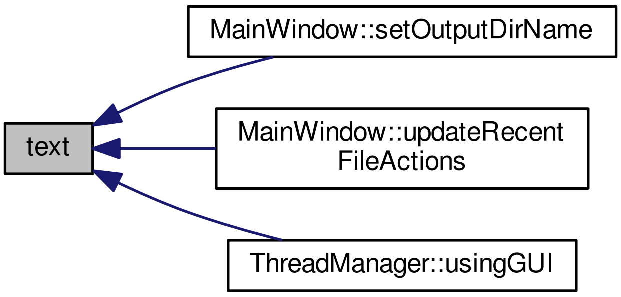
3.5.1.12 def output_parser_lib.text (*cat*, *text_h*)

Definition at line 849 of file output_parser_lib.py.

Referenced by MainWindow.setOutputDirName(), MainWindow.updateRecentFileActions(), and ThreadManager.usingGUI().

```
00849 def text(cat, text_h):
00850   filename = ""
00851   if cat == ’t’:
00852     filename = g.tableoutdir+’/’+g.tablesmaster+’.tex’
00853   elif cat == ’c’:
00854     filename = g.chartoutdir+’/’+g.chartsmaster+’.tex’
00855   else:
00856     print ("Error in text: not know where to print the title !")
00857     exit(1)
00858   file = open(filename,’a’)
00859   file.write(text_h+"\n")
00860   file.close()
00861 
00862 # ============================================================================
```

Here is the caller graph for this function:


MainWindow::setOutputDirName
text
MainWindow::updateRecent FileActions
ThreadManager::usingGUI

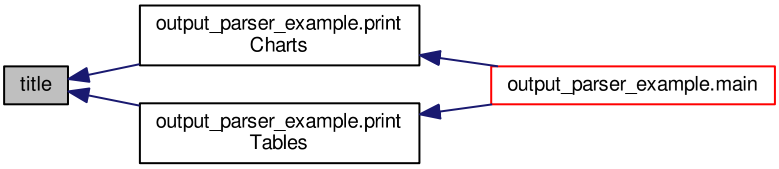
3.5.1.13 def output_parser_lib.title (*cat*, *level*, *title*)

Definition at line 833 of file output_parser_lib.py.

Referenced by output_parser_example.printCharts(), and output_parser_example.printTables().

```
def title (cat, level, title):
  filename = ""
  if cat == 't':
    filename = g.tableoutdir+'/'+g.tablesmaster+'.tex'
  elif cat == 'c':
    filename = g.chartoutdir+'/'+g.chartsmaster+'.tex'
  else:
    print ("Error in printTable: not know where to print the title !")
    exit(1)
  file = open(filename,'a')

  file.write("\n\\clearpage\n")
  file.write("\\"+level+"{"+title+"}\n")
  file.close()

# ============================================================================
```

Here is the caller graph for this function:


title
output_parser_example.print Charts
output_parser_example.print Tables
output_parser_example.main

3.6 Ui Namespace Reference

Classes

- class MainWindow

4 Class Documentation

4.1 AnyOption Class Reference

`#include <anyoption.h>`

Public Member Functions

- AnyOption ()
- AnyOption (int maxoptions)
- AnyOption (int maxoptions, int maxcharoptions)
- ∼AnyOption ()
- void setCommandPrefixChar (char _prefix)
- void setCommandLongPrefix (char ∗_prefix)
- void setFileCommentChar (char _comment)
- void setFileDelimiterChar (char _delimiter)
- void useCommandArgs (int _argc, char ∗∗_argv)
- void useFiileName (const char ∗_filename)

- void noPOSIX ()
- void setVerbose ()
- void setOption (const char ∗opt_string)
- void setOption (char opt_char)
- void setOption (const char ∗opt_string, char opt_char)
- void setFlag (const char ∗opt_string)
- void setFlag (char opt_char)
- void setFlag (const char ∗opt_string, char opt_char)
- void setCommandOption (const char ∗opt_string)
- void setCommandOption (char opt_char)
- void setCommandOption (const char ∗opt_string, char opt_char)
- void setCommandFlag (const char ∗opt_string)
- void setCommandFlag (char opt_char)
- void setCommandFlag (const char ∗opt_string, char opt_char)
- void setFileOption (const char ∗opt_string)
- void setFileOption (char opt_char)
- void setFileOption (const char ∗opt_string, char opt_char)
- void setFileFlag (const char ∗opt_string)
- void setFileFlag (char opt_char)
- void setFileFlag (const char ∗opt_string, char opt_char)
- void processOptions ()
- void processCommandArgs ()
- void processCommandArgs (int max_args)
- bool processFile ()
- void processCommandArgs (int _argc, char ∗∗_argv)
- void processCommandArgs (int _argc, char ∗∗_argv, int max_args)
- bool processFile (const char ∗_filename)
- char ∗ getValue (const char ∗_option)
- bool getFlag (const char ∗_option)
- char ∗ getValue (char _optchar)
- bool getFlag (char _optchar)
- void printUsage ()
- void printAutoUsage ()
- void addUsage (const char ∗line)
- void printHelp ()
- void autoUsagePrint (bool flag)
- int getArgc ()
- char ∗ getArgv (int index)
- bool hasOptions ()

Private Member Functions

- void init ()
- void init (int maxopt, int maxcharopt)
- bool alloc ()
- void cleanup ()
- bool valueStoreOK ()
- bool doubleOptStorage ()
- bool doubleCharStorage ()
- bool doubleUsageStorage ()
- bool setValue (const char ∗option, char ∗value)
- bool setFlagOn (const char ∗option)
- bool setValue (char optchar, char ∗value)
- bool setFlagOn (char optchar)

- void addOption (const char ∗option, int type)
- void addOption (char optchar, int type)
- void addOptionError (const char ∗opt)
- void addOptionError (char opt)
- bool findFlag (char ∗value)
- void addUsageError (const char ∗line)
- bool CommandSet ()
- bool FileSet ()
- bool POSIX ()
- char parsePOSIX (char ∗arg)
- int parseGNU (char ∗arg)
- bool matchChar (char c)
- int matchOpt (char ∗opt)
- char ∗ readFile ()
- char ∗ readFile (const char ∗fname)
- bool consumeFile (char ∗buffer)
- void processLine (char ∗theline, int length)
- char ∗ chomp (char ∗str)
- void valuePairs (char ∗type, char ∗value)
- void justValue (char ∗value)
- void printVerbose (const char ∗msg)
- void printVerbose (char ∗msg)
- void printVerbose (char ch)
- void printVerbose ()

Private Attributes

- int argc
- char ∗∗ argv
- const char ∗ filename
- char ∗ appname
- int ∗ new_argv
- int new_argc
- int max_legal_args
- int max_options
- const char ∗∗ options
- int ∗ optiontype
- int ∗ optionindex
- int option_counter
- int max_char_options
- char ∗ optionchars
- int ∗ optchartype
- int ∗ optcharindex
- int optchar_counter
- char ∗∗ values
- int g_value_counter
- const char ∗∗ usage
- int max_usage_lines
- int usage_lines
- bool command_set
- bool file_set
- bool mem_allocated
- bool posix_style
- bool verbose

- bool print_usage
- bool print_help
- char opt_prefix_char
- char long_opt_prefix [MAX_LONG_PREFIX_LENGTH+1]
- char file_delimiter_char
- char file_comment_char
- char equalsign
- char comment
- char delimiter
- char endofline
- char whitespace
- char nullterminate
- bool set
- bool once
- bool hasoptions
- bool autousage

4.1.1 Detailed Description

Definition at line 32 of file anyoption.h.

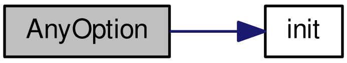
4.1.2 Constructor & Destructor Documentation

4.1.2.1 AnyOption ( )

Definition at line 65 of file anyoption.cpp.

```
00066 {
00067   init();
00068 }
```

Here is the call graph for this function:

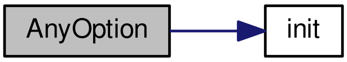


4.1.2.2 AnyOption (int *maxoptions*)

Definition at line 70 of file anyoption.cpp.

```
00071 {
00072   init( maxopt , maxopt );
00073 }
```

Here is the call graph for this function:

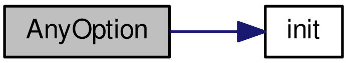


4.1.2.3 AnyOption (int *maxoptions*, int *maxcharoptions*)

Definition at line 75 of file anyoption.cpp.

```
00076 {
00077   init( maxopt , maxcharopt );
00078 }
```

Here is the call graph for this function:

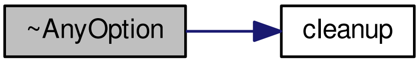


4.1.2.4 ∼AnyOption ( )

Definition at line 80 of file anyoption.cpp.

```
00081 {
00082   if( mem_allocated )
00083     cleanup();
00084 }
```

Here is the call graph for this function:



4.1.3 Member Function Documentation

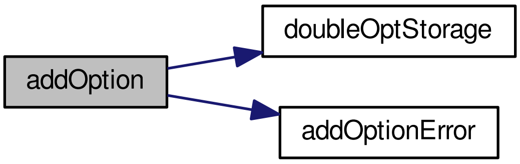
4.1.3.1 void addOption (const char ∗ *option*, int *type*) [private]

Definition at line 521 of file anyoption.cpp.

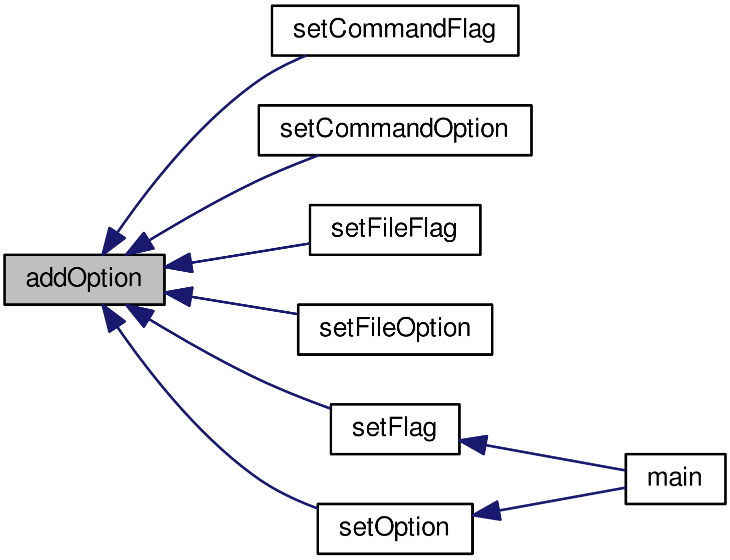
Referenced by setCommandFlag(), setCommandOption(), setFileFlag(), setFileOption(), setFlag(), and setOption().

```
{
  if( option_counter >= max_options ){
    if( doubleOptStorage() == false ){
      addOptionError( opt );
      return;
    }
  }
  options[ option_counter ] = opt ;
  optiontype[ option_counter ] =  type ;
  optionindex[ option_counter ] = g_value_counter;
  option_counter++;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

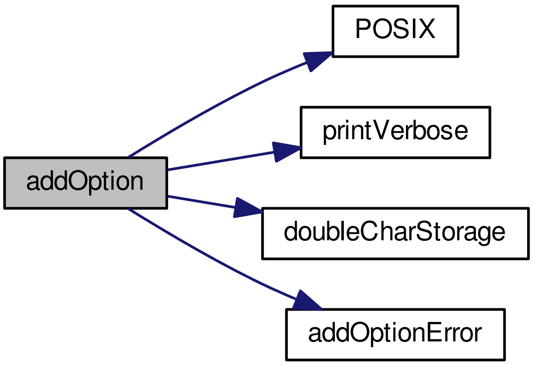


4.1.3.2 void addOption (char *optchar,* int *type*) [private]

Definition at line 536 of file anyoption.cpp.

```
{
  if( !POSIX() ){
    printVerbose("Ignoring the option character \"");
    printVerbose(  opt );
    printVerbose( "\" ( POSIX options are turned off )" );
    printVerbose();
    return;
  }


  if( optchar_counter >= max_char_options ){
    if( doubleCharStorage() == false ){
      addOptionError( opt );
      return;
    }
  }
  optionchars[ optchar_counter ] =  opt ;
  optchartype[ optchar_counter ] =  type ;
  optcharindex[ optchar_counter ] = g_value_counter;
  optchar_counter++;
}
```

Here is the call graph for this function:



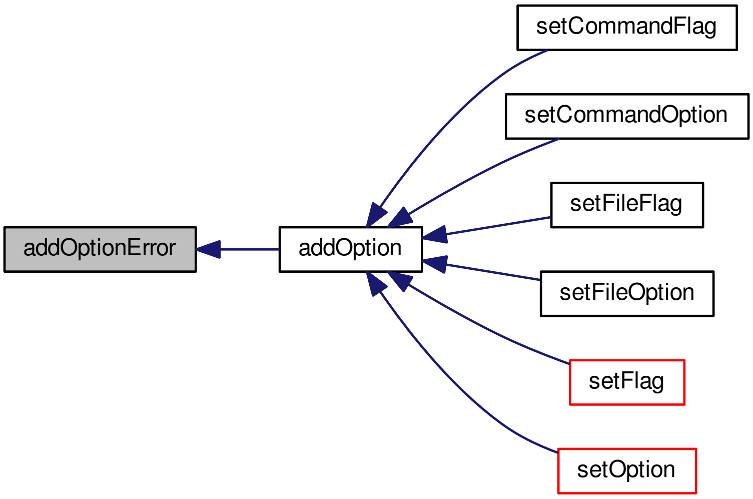
4.1.3.3 void addOptionError (const char ∗ *opt*) [private]

Definition at line 560 of file anyoption.cpp.

Referenced by addOption().

```
{
  cout << endl ;
  cout << "OPTIONS ERROR : Failed allocating extra memory " << endl ;
  cout << "While adding the option : \""<< opt << "\"" << endl;
  cout << "Exiting." << endl ;
  cout << endl ;
  exit(0);
}
```

Here is the caller graph for this function:



4.1.3.4 void addOptionError (char *opt*) [private]

Definition at line 571 of file anyoption.cpp.

```
{
  cout << endl ;
  cout << "OPTIONS ERROR : Failed allocating extra memory " << endl ;
  cout << "While adding the option: \""<< opt << "\"" << endl;
  cout << "Exiting." << endl ;
  cout << endl ;
  exit(0);
}
```

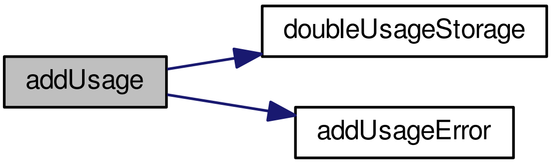
4.1.3.5 void addUsage (const char ∗ *line*)

Definition at line 1153 of file anyoption.cpp.

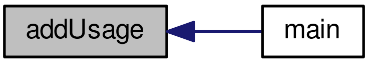
Referenced by main().

```
{
  if( usage_lines >= max_usage_lines ){
    if( doubleUsageStorage() == false ){
      addUsageError( line );
      exit(1);
    }
  }
  usage[ usage_lines ] = line ;
  usage_lines++;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



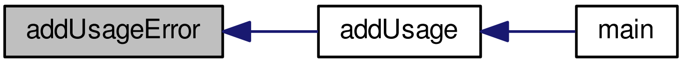
4.1.3.6 void addUsageError (const char ∗ *line*) [private]

Definition at line 1166 of file anyoption.cpp.

Referenced by addUsage().

```
{
  cout << endl ;
  cout << "OPTIONS ERROR : Failed allocating extra memory " << endl ;
  cout << "While adding the usage/help  : \""<< line << "\"" << endl;
  cout << "Exiting." << endl ;
  cout << endl ;
  exit(0);

}
```

Here is the caller graph for this function:



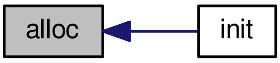
4.1.3.7 bool alloc ( ) [private]

Definition at line 143 of file anyoption.cpp.

Referenced by init().

```
{
  int i = 0 ;
  int size = 0 ;

  if( mem_allocated )
    return true;

  size = (max_options+1) * sizeof(const char*);
  options = (const char**)malloc( size );
  optiontype = (int*) malloc( (max_options+1)*sizeof(int) );
  optionindex = (int*) malloc( (max_options+1)*sizeof(int) );
  if( options == NULL || optiontype == NULL || optionindex == NULL )
    return false;
  else
    mem_allocated  = true;
  for( i = 0 ; i < max_options ; i++ ){
    options[i] = NULL;
    optiontype[i] = 0 ;
    optionindex[i] = -1 ;
  }
  optionchars = (char*) malloc( (max_char_options+1)*sizeof(char) );
  optchartype = (int*) malloc( (max_char_options+1)*sizeof(int) );
  optcharindex = (int*) malloc( (max_char_options+1)*sizeof(int) );
  if( optionchars == NULL ||
            optchartype == NULL ||
            optcharindex == NULL )
        {
    mem_allocated = false;
    return false;
  }
  for( i = 0 ; i < max_char_options ; i++ ){
    optionchars[i] = '0';
    optchartype[i] = 0 ;
    optcharindex[i] = -1 ;
  }

  size = (max_usage_lines+1) * sizeof(const char*) ;
  usage = (const char**) malloc( size );

  if( usage == NULL  ){
    mem_allocated = false;
    return false;
  }
  for( i = 0 ; i < max_usage_lines ; i++ )
    usage[i] = NULL;

  return true;
}
```

Here is the caller graph for this function:



4.1.3.8 void autoUsagePrint (bool *flag*)

Definition at line 362 of file anyoption.cpp.

```
{
  autousage = _autousage;
}
```

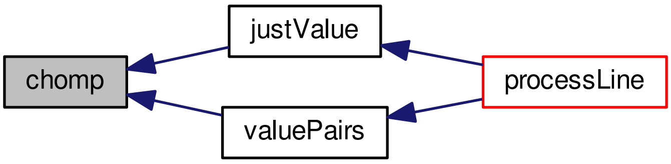
4.1.3.9 char * chomp (char * *str*) [private]

Definition at line 1053 of file anyoption.cpp.

Referenced by justValue(), and valuePairs().

```
{
        while( *str == whitespace )
                str++;
        char *end = str+strlen(str)-1;
        while( *end == whitespace )
                end--;
        *(end+1) = nullterminate;
        return str;
}
```

Here is the caller graph for this function:



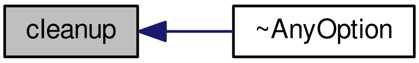
4.1.3.10 void cleanup ( ) [private]

Definition at line 253 of file anyoption.cpp.

Referenced by ~AnyOption().

```
{
  free (options);
  free (optiontype);
  free (optionindex);
  free (optionchars);
  free (optchartype);
  free (optcharindex);
  free (usage);
  if( values != NULL )
    free (values);
  if( new_argv != NULL )
    free (new_argv);
}
```

Here is the caller graph for this function:



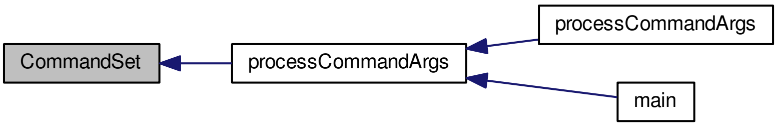
4.1.3.11 bool CommandSet ( ) [private]

Definition at line 298 of file anyoption.cpp.

Referenced by processCommandArgs().

```
{
  return( command_set );
}
```

Here is the caller graph for this function:



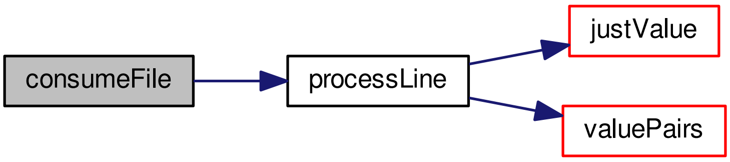
4.1.3.12 bool consumeFile (char * *buffer*) [private]

Definition at line 971 of file anyoption.cpp.

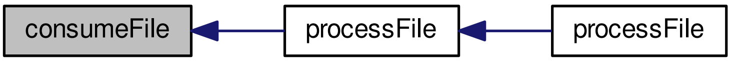
Referenced by processFile().

```
{

        if( buffer == NULL )
    return false;

         char *cursor = buffer;/* preserve the ptr */
         char *pline = NULL ;
         int linelength = 0;
         bool newline = true;
         for( unsigned int i = 0 ; i < strlen( buffer ) ; i++ ){
         if( *cursor == endofline ) { /* end of line */
            if( pline != NULL ) /* valid line */
                   processLine( pline, linelength );
                   pline = NULL;
                   newline = true;
             }else if( newline ){ /* start of line */
                   newline = false;
                  if( (*cursor != comment ) ){ /* not a comment */
            pline = cursor ;
                        linelength = 0 ;
                  }
               }
              cursor++; /* keep moving */
              linelength++;
         }
       free (buffer);
  return true;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



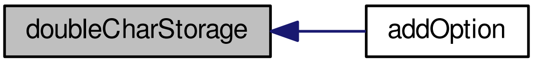
4.1.3.13 bool doubleCharStorage ( ) [private]

Definition at line 215 of file anyoption.cpp.

Referenced by addOption().

```
{
  optionchars = (char*) realloc( optionchars,
      ((2*max_char_options)+1)*sizeof(char) );
  optchartype = (int*) realloc( optchartype,
      ((2*max_char_options)+1)*sizeof(int) );
  optcharindex = (int*) realloc( optcharindex,
      ((2*max_char_options)+1)*sizeof(int) );
  if( optionchars == NULL ||
      optchartype == NULL ||
      optcharindex == NULL )
    return false;
  /* init new storage */
  for( int i = max_char_options ; i < 2*max_char_options ; i++ ){
    optionchars[i] = ’0’;
    optchartype[i] = 0 ;
    optcharindex[i] = -1 ;
  }
  max_char_options = 2 * max_char_options;
  return true;
}
```

Here is the caller graph for this function:



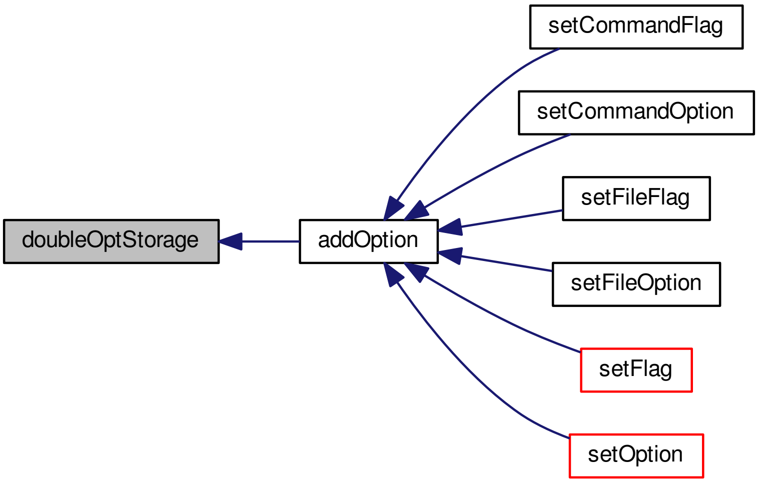
4.1.3.14 bool doubleOptStorage ( ) [private]

Definition at line 194 of file anyoption.cpp.

Referenced by addOption().

```
{
  options = (const char**)realloc( options,
      ((2*max_options)+1) * sizeof( const char*) );
  optiontype = (int*) realloc(  optiontype ,
      ((2 * max_options)+1)* sizeof(int) );
  optionindex = (int*) realloc(  optionindex,
      ((2 * max_options)+1) * sizeof(int) );
  if( options == NULL || optiontype == NULL || optionindex == NULL )
    return false;
  /* init new storage */
  for( int i = max_options ; i < 2*max_options ; i++ ){
    options[i] = NULL;
    optiontype[i] = 0 ;
    optionindex[i] = -1 ;
  }
  max_options = 2 * max_options ;
  return true;
}
```

Here is the caller graph for this function:



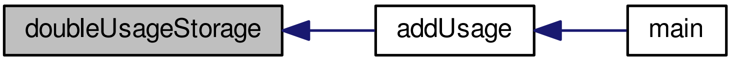
4.1.3.15 bool doubleUsageStorage ( ) [private]

Definition at line 238 of file anyoption.cpp.

Referenced by addUsage().

```
{
  usage = (const char**)realloc( usage,
      ((2*max_usage_lines)+1) * sizeof( const char*) );
  if ( usage == NULL )
    return false;
  for( int i = max_usage_lines ; i < 2*max_usage_lines ; i++ )
    usage[i] = NULL;
  max_usage_lines = 2 * max_usage_lines ;
  return true;

}
```

Here is the caller graph for this function:



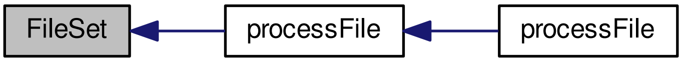
4.1.3.16 bool FileSet ( ) [private]

Definition at line 304 of file anyoption.cpp.

Referenced by processFile().

```
00305 {
00306   return( file_set );
00307 }
```

Here is the caller graph for this function:



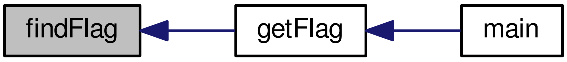
4.1.3.17 bool findFlag (char ∗ *value*) [private]

Definition at line 829 of file anyoption.cpp.

Referenced by getFlag().

```
00830 {
00831   if( val == NULL )
00832     return false;
00833
00834   if( strcmp( TRUE_FLAG , val ) == 0 )
00835     return true;
00836
00837   return false;
00838 }
```

Here is the caller graph for this function:



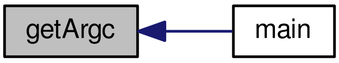
4.1.3.18 int getArgc ( )

Definition at line 905 of file anyoption.cpp.

Referenced by main().

```
{
  return new_argc;
}
```

Here is the caller graph for this function:



4.1.3.19 char ∗ getArgv (int *index*)

Definition at line 911 of file anyoption.cpp.

```
{
  if( index < new_argc ){
    return ( argv[ new_argv[ index ] ] );
  }
  return NULL;
}
```

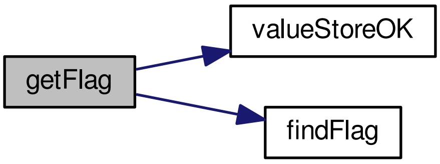
4.1.3.20 bool getFlag (const char ∗ *_option*)

Definition at line 793 of file anyoption.cpp.

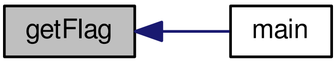
Referenced by main().

```
{
  if( !valueStoreOK() )
    return false;
  for( int i = 0 ; i < option_counter ; i++ ){
    if( strcmp( options[i], option ) == 0 )
      return findFlag( values[ optionindex[i] ] );
  }
  return false;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

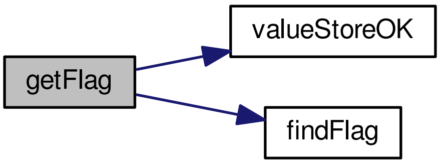


4.1.3.21 bool getFlag (char _*optchar*)

Definition at line 817 of file anyoption.cpp.

```
{
  if( !valueStoreOK() )
    return false;
  for( int i = 0 ; i < optchar_counter ; i++ ){
    if( optionchars[i] == option )
      return findFlag( values[ optcharindex[i] ] ) ;
  }
  return false;
}
```

Here is the call graph for this function:



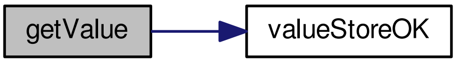
4.1.3.22 char ∗ getValue (const char ∗ *_option*)

Definition at line 780 of file anyoption.cpp.

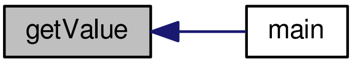
Referenced by main().

```
{
  if( !valueStoreOK() )
    return NULL;

  for( int i = 0 ; i < option_counter ; i++ ){
    if( strcmp( options[i], option ) == 0 )
      return values[ optionindex[i] ];
  }
  return NULL;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

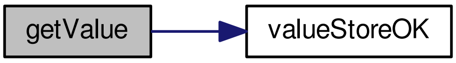


4.1.3.23 char ∗ getValue (char *_optchar*)

Definition at line 805 of file anyoption.cpp.

```
{
  if( !valueStoreOK() )
    return NULL;
  for( int i = 0 ; i < optchar_counter ; i++ ){
    if( optionchars[i] == option )
      return values[ optcharindex[i] ];
  }
  return NULL;
}
```

Here is the call graph for this function:



4.1.3.24 bool hasOptions ( )

Definition at line 356 of file anyoption.cpp.

```
{
  return hasoptions;
}
```

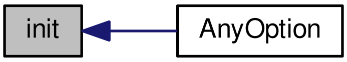
4.1.3.25 void init ( ) [private]

Definition at line 87 of file anyoption.cpp.

Referenced by AnyOption().

```
{
  init( DEFAULT_MAXOPTS , DEFAULT_MAXOPTS );
}
```

Here is the caller graph for this function:

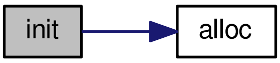


4.1.3.26 void init (int *maxopt*, int *maxcharopt*) [private]

Definition at line 93 of file anyoption.cpp.

```
{

  max_options    = maxopt;
  max_char_options = maxcharopt;
  max_usage_lines  = DEFAULT_MAXUSAGE;
  usage_lines  = 0 ;
  argc     = 0;
  argv     = NULL;
  posix_style  = true;
  verbose   = false;
  filename   = NULL;
  appname   = NULL;
  option_counter   = 0;
  optchar_counter  = 0;
  new_argv   = NULL;
  new_argc   = 0 ;
  max_legal_args   = 0 ;
  command_set   = false;
  file_set   = false;
  values     = NULL;
  g_value_counter = 0;
  mem_allocated   = false;
  command_set   = false;
  file_set  = false;
  opt_prefix_char     = '-';
  file_delimiter_char = ':';
  file_comment_char   = '#';
  equalsign   = '=';
  comment       = '#' ;
  delimiter     = ':' ;
  endofline     = '\n';
  whitespace    = ' ' ;
  nullterminate = '\0';
  set = false;
  once = true;
  hasoptions = false;
  autousage = false;

  strcpy( long_opt_prefix , "--" );

  if( alloc() == false ){
    cout << endl << "OPTIONS ERROR : Failed allocating memory" ;
    cout << endl ;
    cout << "Exiting." << endl ;
    exit (0);
  }
}
```

Here is the call graph for this function:



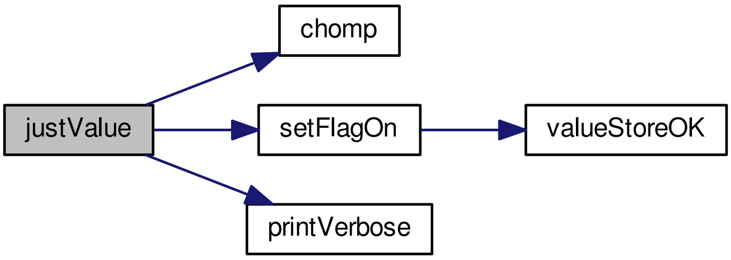
4.1.3.27 void justValue (char * *value*) [private]

Definition at line 1096 of file anyoption.cpp.

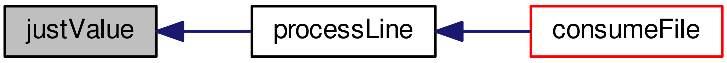
Referenced by processLine().

```
{

  if ( strlen(chomp(type)) == 1  ){ /* this is a char option */
    for( int i = 0 ; i < optchar_counter ; i++ ){
      if(  optionchars[i] == type[0]  ){ /* match */
        if( optchartype[i] == COMMON_FLAG ||
            optchartype[i] == FILE_FLAG )
        {
           setFlagOn( type[0] );
           return;
        }
      }
    }
  }
  /* if no char options matched */
  for( int i = 0 ; i < option_counter ; i++ ){
    if( strcmp( options[i], type ) == 0 ){ /* match */
      if( optiontype[i] == COMMON_FLAG ||
          optiontype[i] == FILE_FLAG )
      {
         setFlagOn( type );
         return;
      }
    }
  }
        printVerbose( "Unknown option in resourcefile : " );
  printVerbose( type  );
  printVerbose( );
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



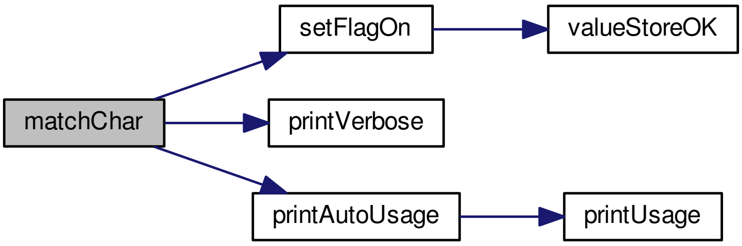
4.1.3.28 bool matchChar (char *c*) [private]

Definition at line 738 of file anyoption.cpp.

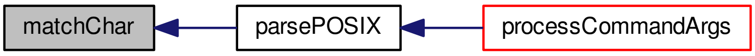
Referenced by parsePOSIX().

```
00739 {
00740   for( int i = 0 ; i < optchar_counter ; i++ ){
00741     if( optionchars[i] == c ) { /* found match */
00742       if(optchartype[i] == COMMON_OPT ||
00743            optchartype[i] == COMMAND_OPT )
00744       { /* an option store and stop scanning */
00745         return true;
00746       }else if( optchartype[i] == COMMON_FLAG ||
00747           optchartype[i] == COMMAND_FLAG ) { /* a flag store and keep scanning */
00748         setFlagOn( c );
00749         return false;
00750       }
00751     }
00752   }
00753   printVerbose( "Unknown command argument option : " );
00754   printVerbose( c ) ;
00755   printVerbose( );
00756   printAutoUsage();
00757   return false;
00758 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



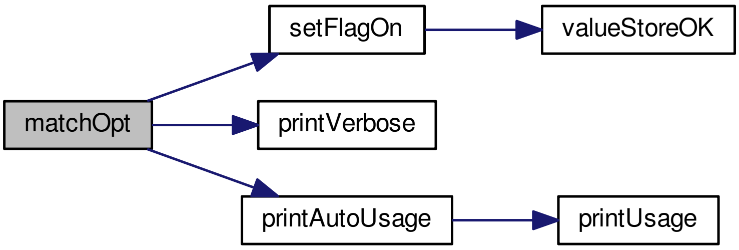
4.1.3.29 int matchOpt (char ∗ *opt*) [private]

Definition at line 715 of file anyoption.cpp.

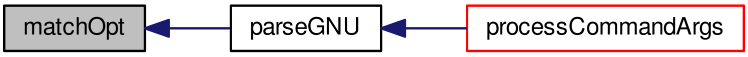
Referenced by parseGNU().

```
00716 {
00717   for( int i = 0 ; i < option_counter ; i++ ){
00718     if( strcmp( options[i], opt ) == 0 ){
00719       if( optiontype[i] ==  COMMON_OPT ||
00720           optiontype[i] ==  COMMAND_OPT )
00721       { /* found option return index */
00722         return i;
00723       }else if( optiontype[i] == COMMON_FLAG ||
00724              optiontype[i] == COMMAND_FLAG )
00725       { /* found flag, set it */
00726         setFlagOn( opt );
00727         return -1;
00728       }
00729     }
00730   }
00731   printVerbose( "Unknown command argument option : " );
00732   printVerbose( opt  ) ;
00733   printVerbose( );
00734   printAutoUsage();
00735   return  -1;
00736 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.3.30 void noPOSIX ( )

Definition at line 310 of file anyoption.cpp.

```
00311 {
00312   posix_style = false;
00313 }
```

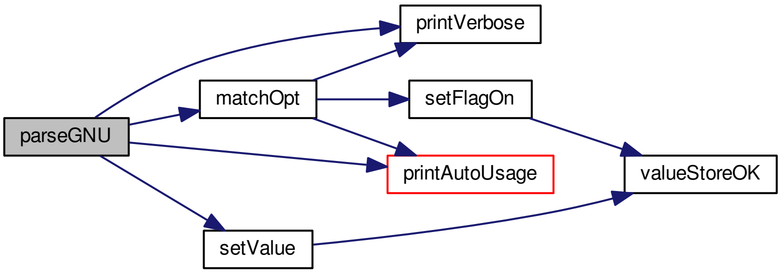
4.1.3.31 int parseGNU (char ∗ *arg*) `[private]`

Definition at line 680 of file anyoption.cpp.

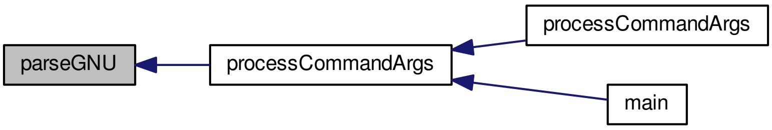
Referenced by processCommandArgs().

```
{
  int split_at = 0;
  /* if has a '=' sign get value */
  for( unsigned int i = 0 ; i < strlen(arg) ; i++ ){
    if(arg[i] ==  equalsign ){
      split_at = i ; /* store index */
      i = strlen(arg); /* get out of loop */
    }
  }
  if( split_at > 0 ){ /* it is an option value pair */
    char* tmp = (char*) malloc(  (split_at+1)*sizeof(char) );
    for( int i = 0 ; i < split_at ; i++ )
      tmp[i] = arg[i];
    tmp[split_at] = '\0';

    if ( matchOpt( tmp ) >= 0 ){
      setValue( options[matchOpt(tmp)] , arg+split_at+1 );
      free (tmp);
    }else{
      printVerbose( "Unknown command argument option : " );
      printVerbose( arg );
      printVerbose( );
      printAutoUsage();
      free (tmp);
      return -1;
    }
  }else{ /* regular options with no '=' sign  */
    return  matchOpt(arg);
  }
  return -1;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



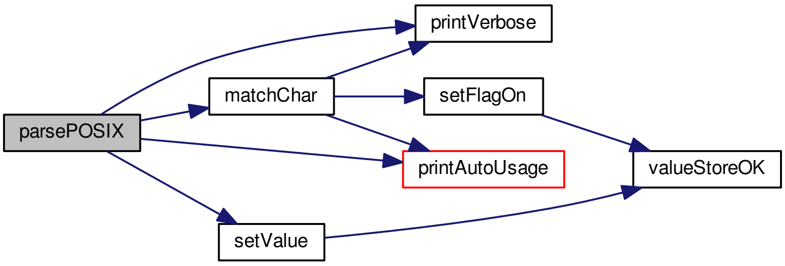
4.1.3.32 char parsePOSIX (char ∗ *arg*) [private]

Definition at line 653 of file anyoption.cpp.

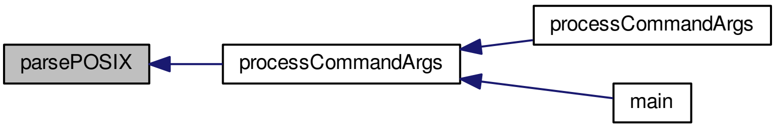
Referenced by processCommandArgs().

```
{

  for( unsigned int i = 0 ; i < strlen(arg) ; i++ ){
    char ch = arg[i] ;
    if( matchChar(ch) ) { /* keep matching flags till an option */
      /*if last char argv[++i] is the value */
      if( i == strlen(arg)-1 ){
        return ch;
      }else{/* else the rest of arg is the value */
        i++; /* skip any '=' and ' ' */
        while( arg[i] == whitespace
              || arg[i] == equalsign )
          i++;
        setValue( ch , arg+i );
        return '0';
      }
    }
  }
  printVerbose( "Unknown command argument option : " );
  printVerbose( arg );
  printVerbose( );
  printAutoUsage();
  return '0';
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



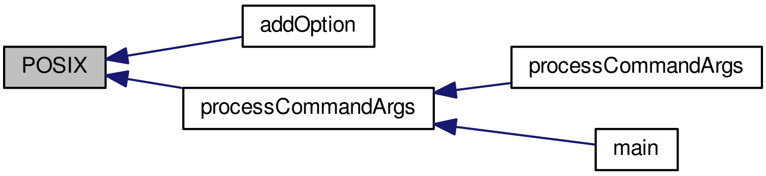
4.1.3.33 bool POSIX ( ) `[private]`

Definition at line 316 of file anyoption.cpp.

Referenced by addOption(), and processCommandArgs().

```
{
  return posix_style;
}
```

Here is the caller graph for this function:



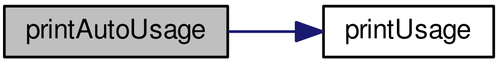
4.1.3.34 void printAutoUsage ( )

Definition at line 1133 of file anyoption.cpp.

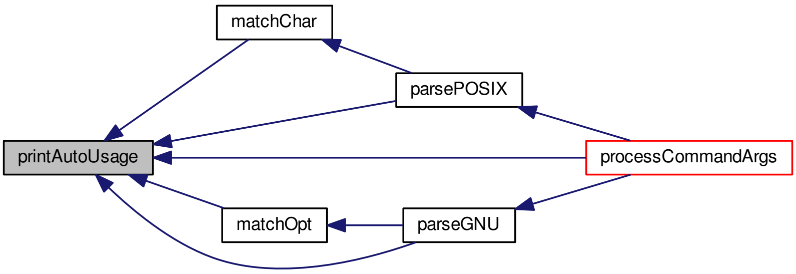
Referenced by matchChar(), matchOpt(), parseGNU(), parsePOSIX(), and processCommandArgs().

```
01134 {
01135   if( autousage ) printUsage();
01136 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.1.3.35 void printHelp ( )

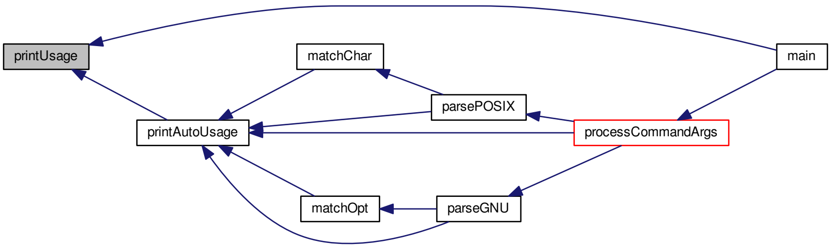
4.1.3.36 void printUsage ( )

Definition at line 1139 of file anyoption.cpp.

Referenced by main(), and printAutoUsage().

```
{

  if( once ) {
    once = false ;
    cout << endl ;
    for( int i = 0 ; i < usage_lines ; i++ )
      cout << usage[i] << endl ;
    cout << endl ;
  }
}
```

Here is the caller graph for this function:



4.1.3.37 void printVerbose (const char ∗ *msg*) `[private]`

Definition at line 335 of file anyoption.cpp.

```
{
  if( verbose )
    cout << msg  ;
}
```

4.1.3.38 void printVerbose (char ∗ *msg*) `[private]`

Definition at line 342 of file anyoption.cpp.

```
{
  if( verbose )
    cout << msg  ;
}
```

4.1.3.39 void printVerbose (char *ch*) `[private]`

Definition at line 349 of file anyoption.cpp.

```
{
  if( verbose )
    cout << ch ;
}
```

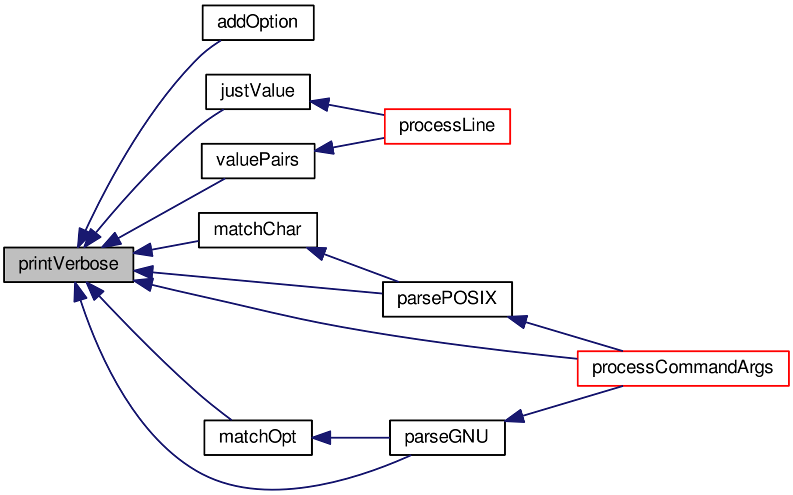
4.1.3.40 void printVerbose ( ) [private]

Definition at line 329 of file anyoption.cpp.

Referenced by addOption(), justValue(), matchChar(), matchOpt(), parseGNU(), parsePOSIX(), processCommandArgs(), and valuePairs().

```
{
  if( verbose )
    cout << endl  ;
}
```

Here is the caller graph for this function:



4.1.3.41 void processCommandArgs ( )

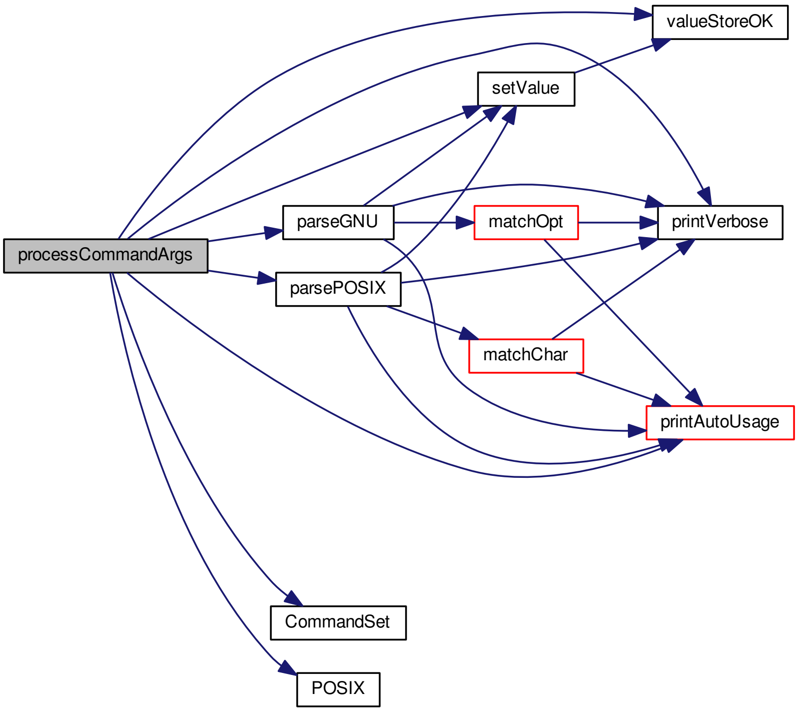
Definition at line 610 of file anyoption.cpp.

Referenced by main(), and processCommandArgs().

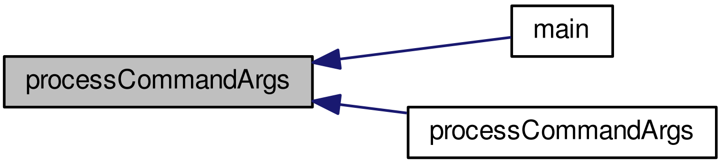
```
{
     if( ! ( valueStoreOK() && CommandSet() )  )
     return;

  if( max_legal_args == 0 )
    max_legal_args = argc;
  new_argv = (int*) malloc( (max_legal_args+1) * sizeof(int) );
  for( int i = 1 ; i < argc ; i++ ){/* ignore first argv */
    if(  argv[i][0] == long_opt_prefix[0] &&
                     argv[i][1] == long_opt_prefix[1] ) { /* long GNU option */
      int match_at = parseGNU( argv[i]+2 ); /* skip -- */
      if( match_at >= 0 && i < argc-1 ) /* found match */
        setValue( options[match_at] , argv[++i] );
    }else if(  argv[i][0] ==  opt_prefix_char ) { /* POSIX char */
      if( POSIX() ){
        char ch =  parsePOSIX( argv[i]+1 );/* skip - */
        if( ch != '0'  && i < argc-1 ) /* matching char */
          setValue( ch ,  argv[++i] );
      } else { /* treat it as GNU option with a - */
        int match_at = parseGNU( argv[i]+1 ); /* skip - */
        if( match_at >= 0 && i < argc-1 ) /* found match */
```

```
          setValue( options[match_at] , argv[++i] );
      }
    }else { /* not option but an argument keep index */
      if( new_argc < max_legal_args ){
                                new_argv[ new_argc ] = i ;
                                new_argc++;
                        }else{ /* ignore extra arguments */
                                printVerbose( "Ignoring extra argument: " );
        printVerbose( argv[i] );
        printVerbose( );
        printAutoUsage();
                        }
      printVerbose( "Unknown command argument option : " );
      printVerbose( argv[i] );
      printVerbose( );
      printAutoUsage();
    }
  }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

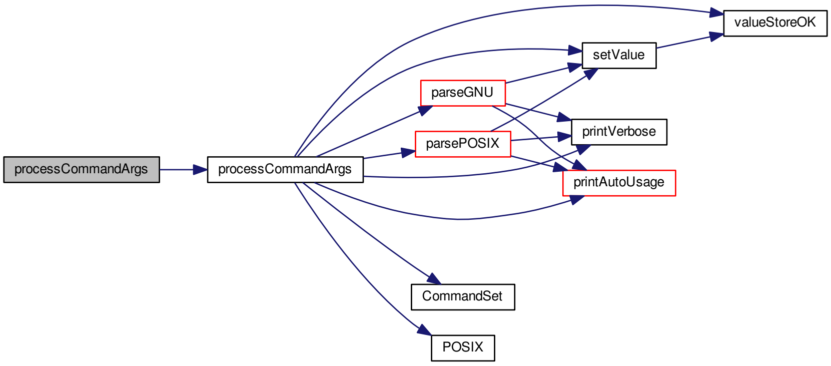


4.1.3.42 void processCommandArgs (int *max_args*)

Definition at line 589 of file anyoption.cpp.

```
00590 {
00591   max_legal_args = max_args;
00592   processCommandArgs();
00593 }
```

Here is the call graph for this function:

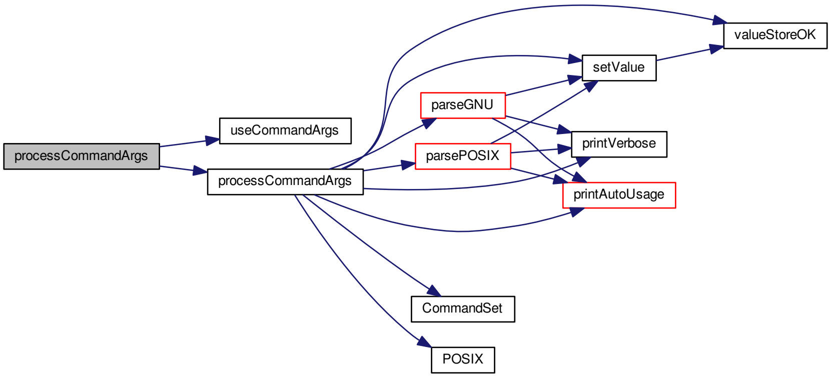


4.1.3.43 void processCommandArgs (int *_argc,* char ∗∗ *_argv*)

Definition at line 603 of file anyoption.cpp.

```
00604 {
00605   useCommandArgs( _argc, _argv );
00606   processCommandArgs();
00607 }
```

Here is the call graph for this function:

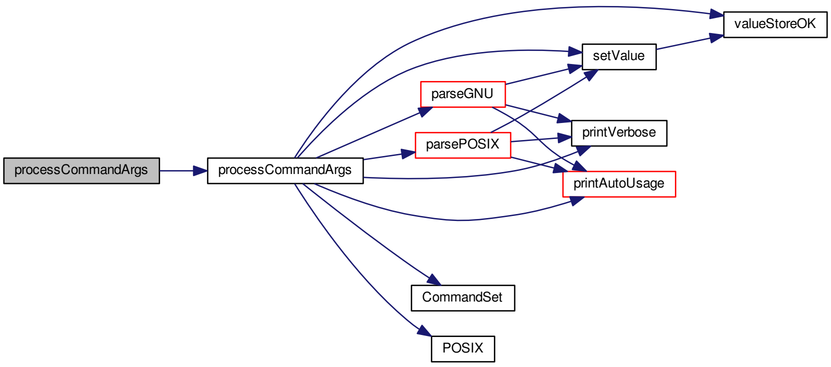


4.1.3.44 void processCommandArgs (int *_argc*, char ∗∗ *_argv*, int *max_args*)

Definition at line 596 of file anyoption.cpp.

```
00597 {
00598   max_legal_args = max_args;
00599   processCommandArgs(  _argc, _argv );
00600 }
```

Here is the call graph for this function:



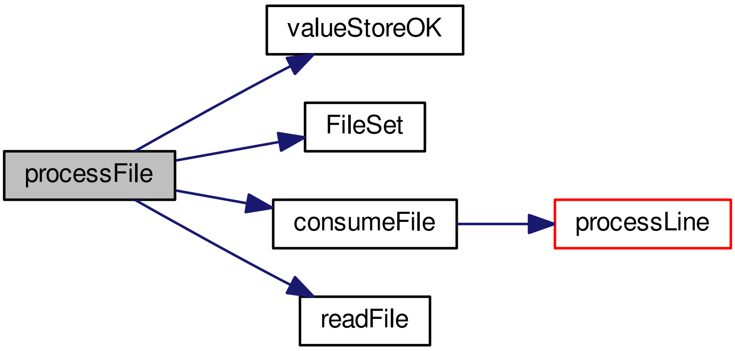
4.1.3.45 bool processFile ( )

Definition at line 922 of file anyoption.cpp.

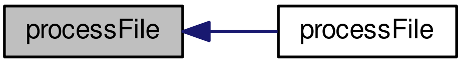
Referenced by processFile().

```
00923 {
00924   if( ! (valueStoreOK() && FileSet())  )
00925     return false;
00926   return  ( consumeFile(readFile()) );
00927 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

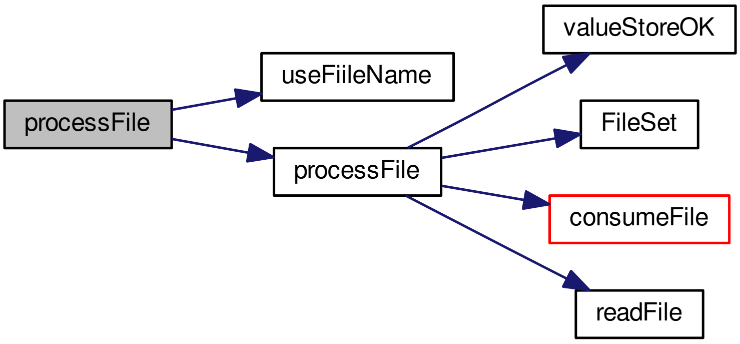


4.1.3.46 bool processFile (const char * *_filename*)

Definition at line 930 of file anyoption.cpp.

```
{
  useFiileName(filename );
  return ( processFile() );
}
```

Here is the call graph for this function:



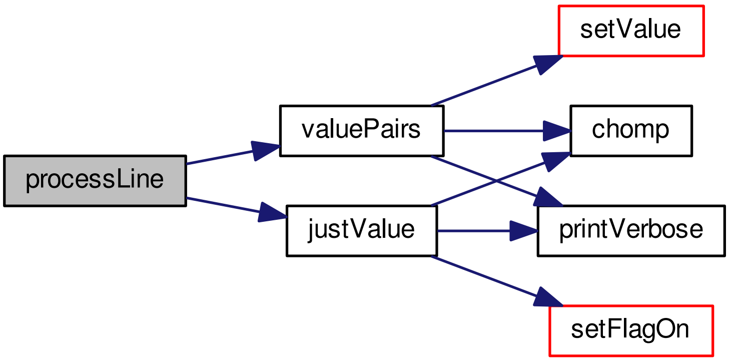
4.1.3.47 void processLine (char * *theline*, int *length*) [private]

Definition at line 1023 of file anyoption.cpp.

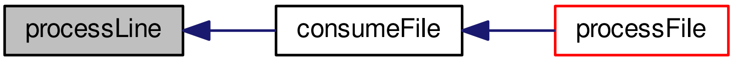
Referenced by consumeFile().

```
{
        bool found = false;
        char *pline = (char*) malloc( (length+1)*sizeof(char) );
        for( int i = 0 ; i < length ; i ++ )
                pline[i]= *(theline++);
        pline[length] = nullterminate;
        char *cursor = pline ; /* preserve the ptr */
        if( *cursor == delimiter || *(cursor+length-1) == delimiter ){
                justValue( pline );/* line with start/end delimiter */
        }else{
                for( int i = 1 ; i < length-1 && !found ; i++){/* delimiter */
                        if( *cursor == delimiter ){
                                *(cursor-1) = nullterminate; /* two strings */
                                found = true;
                                valuePairs( pline , cursor+1 );
                        }
                        cursor++;
                }
                cursor++;
                if( !found ) /* not a pair */
                        justValue( pline );
        }
        free (pline);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

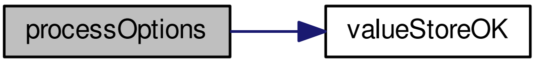


4.1.3.48 void processOptions ( )

Definition at line 582 of file anyoption.cpp.

```
{
  if( ! valueStoreOK() )
    return;
}
```

Here is the call graph for this function:



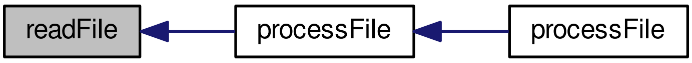
4.1.3.49 char ∗ readFile ( ) [private]

Definition at line 937 of file anyoption.cpp.

Referenced by processFile().

```
{
  return ( readFile(filename) );
}
```

Here is the caller graph for this function:



4.1.3.50 char ∗ readFile (const char ∗ *fname*) [private]

Definition at line 947 of file anyoption.cpp.

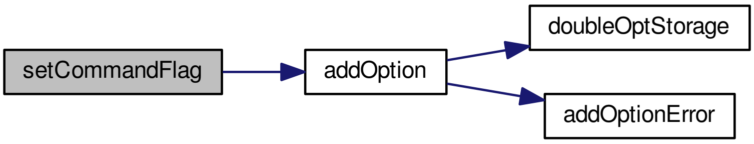
```
{
        int length;
        char *buffer;
        ifstream is;
        is.open ( fname , ifstream::in );
        if( ! is.good() ){
                is.close();
                return NULL;
        }
        is.seekg (0, ios::end);
        length = is.tellg();
        is.seekg (0, ios::beg);
        buffer = (char*) malloc(length*sizeof(char));
        is.read (buffer,length);
        is.close();
        return buffer;
}
```

4.1.3.51 void setCommandFlag (const char ∗ *opt_string*)

Definition at line 411 of file anyoption.cpp.

```
{
  addOption( opt , COMMAND_FLAG );
  g_value_counter++;
}
```

Here is the call graph for this function:

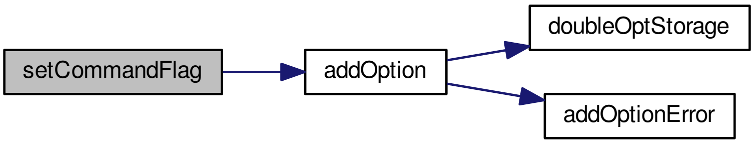


4.1.3.52 void setCommandFlag (char *opt_char*)

Definition at line 418 of file anyoption.cpp.

```
{
  addOption( opt , COMMAND_FLAG );
  g_value_counter++;
}
```

Here is the call graph for this function:

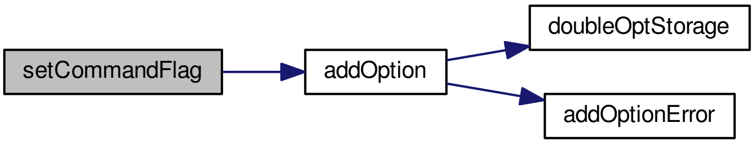


4.1.3.53 void setCommandFlag (const char ∗ *opt_string*, char *opt_char*)

Definition at line 425 of file anyoption.cpp.

```
{
  addOption( opt , COMMAND_FLAG );
  addOption( optchar , COMMAND_FLAG );
  g_value_counter++;
}
```

Here is the call graph for this function:



4.1.3.54 void setCommandLongPrefix (char * *_prefix*)

Definition at line 275 of file anyoption.cpp.

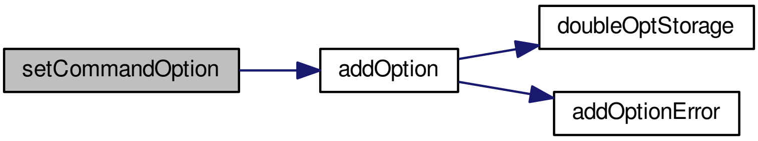
```
{
  if( strlen( _prefix ) > MAX_LONG_PREFIX_LENGTH ){
    *( _prefix + MAX_LONG_PREFIX_LENGTH ) = '\0';
  }

  strcpy (long_opt_prefix,  _prefix);
}
```

4.1.3.55 void setCommandOption (const char * *opt_string*)

Definition at line 389 of file anyoption.cpp.

```
{
  addOption( opt , COMMAND_OPT );
  g_value_counter++;
}
```

Here is the call graph for this function:

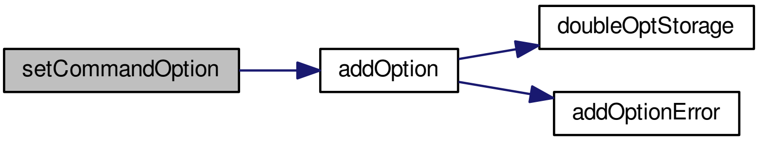


4.1.3.56 void setCommandOption (char *opt_char*)

Definition at line 396 of file anyoption.cpp.

```
00397 {
00398   addOption( opt , COMMAND_OPT );
00399   g_value_counter++;
00400 }
```

Here is the call graph for this function:

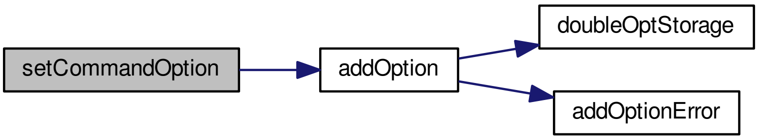


4.1.3.57 void setCommandOption (const char ∗ *opt_string,* char *opt_char*)

Definition at line 403 of file anyoption.cpp.

```
00404 {
00405   addOption( opt , COMMAND_OPT );
00406   addOption( optchar , COMMAND_OPT );
00407   g_value_counter++;
00408 }
```

Here is the call graph for this function:



4.1.3.58 void setCommandPrefixChar (char *_prefix*)

Definition at line 269 of file anyoption.cpp.

```
00270 {
00271   opt_prefix_char = _prefix;
00272 }
```

4.1.3.59 void setFileCommentChar (char *_comment*)

Definition at line 285 of file anyoption.cpp.

```
00286 {
00287   file_delimiter_char = _comment;
00288 }
```

4.1.3.60 void setFileDelimiterChar (char *_delimiter*)

Definition at line 292 of file anyoption.cpp.

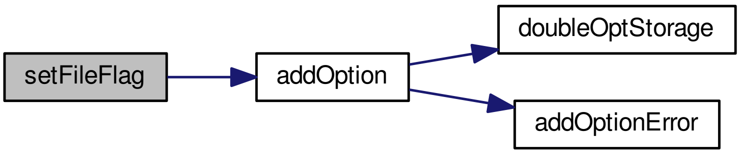
```
00293 {
00294   file_comment_char = _delimiter ;
00295 }
```

4.1.3.61 void setFileFlag (const char ∗ *opt_string*)

Definition at line 455 of file anyoption.cpp.

```
00456 {
00457   addOption( opt , FILE_FLAG );
00458   g_value_counter++;
00459 }
```

Here is the call graph for this function:

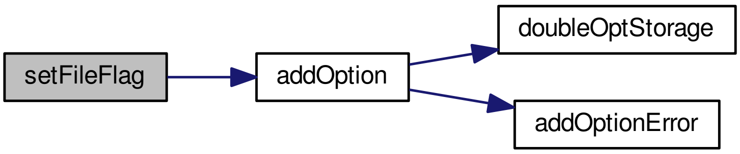


4.1.3.62 void setFileFlag (char *opt_char*)

Definition at line 462 of file anyoption.cpp.

```
00463 {
00464   addOption( opt , FILE_FLAG );
00465   g_value_counter++;
00466 }
```

Here is the call graph for this function:

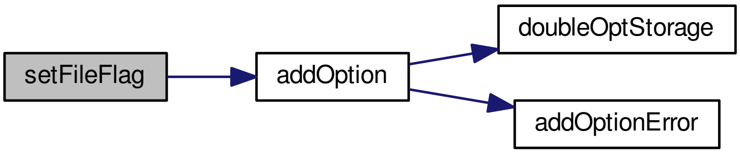


4.1.3.63 void setFileFlag (const char ∗ *opt_string,* char *opt_char*)

Definition at line 469 of file anyoption.cpp.

```
00470 {
00471   addOption( opt , FILE_FLAG );
00472   addOption( optchar , FILE_FLAG );
00473   g_value_counter++;
00474 }
```

Here is the call graph for this function:

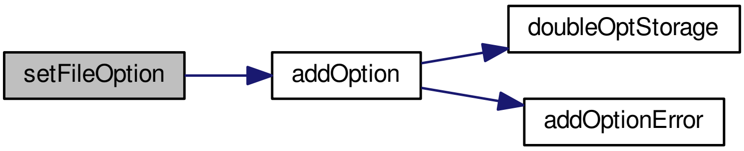


4.1.3.64 void setFileOption (const char ∗ *opt_string*)

Definition at line 433 of file anyoption.cpp.

```
00434 {
00435   addOption( opt , FILE_OPT );
00436   g_value_counter++;
00437 }
```

Here is the call graph for this function:

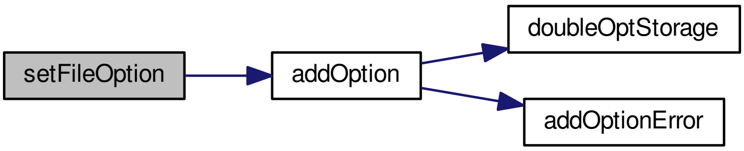


4.1.3.65 void setFileOption (char *opt_char*)

Definition at line 440 of file anyoption.cpp.

```
00441 {
00442   addOption( opt , FILE_OPT );
00443   g_value_counter++;
00444 }
```

Here is the call graph for this function:

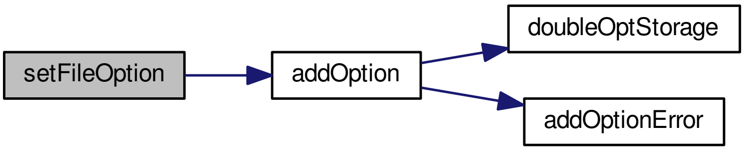


4.1.3.66 void setFileOption (const char ∗ *opt_string,* char *opt_char*)

Definition at line 447 of file anyoption.cpp.

```
00448 {
00449   addOption( opt , FILE_OPT );
00450   addOption( optchar, FILE_OPT  );
00451   g_value_counter++;
00452 }
```

Here is the call graph for this function:



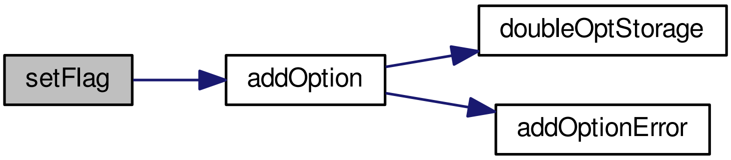
4.1.3.67 void setFlag (const char ∗ *opt_string*)

Definition at line 499 of file anyoption.cpp.

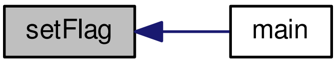
Referenced by main().

```
{
  addOption( opt , COMMON_FLAG );
  g_value_counter++;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

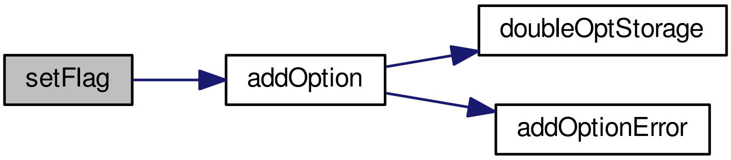


4.1.3.68 void setFlag (char *opt_char*)

Definition at line 506 of file anyoption.cpp.

```
{
  addOption( opt , COMMON_FLAG );
  g_value_counter++;
}
```

Here is the call graph for this function:

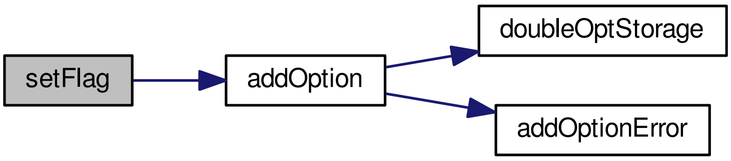


4.1.3.69 void setFlag (const char ∗ *opt_string*, char *opt_char*)

Definition at line 513 of file anyoption.cpp.

```
00514 {
00515   addOption( opt , COMMON_FLAG );
00516   addOption( optchar , COMMON_FLAG );
00517   g_value_counter++;
00518 }
```

Here is the call graph for this function:



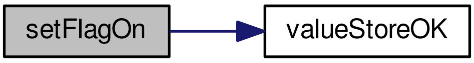
4.1.3.70 bool setFlagOn (const char ∗ *option*) `[private]`

Definition at line 859 of file anyoption.cpp.

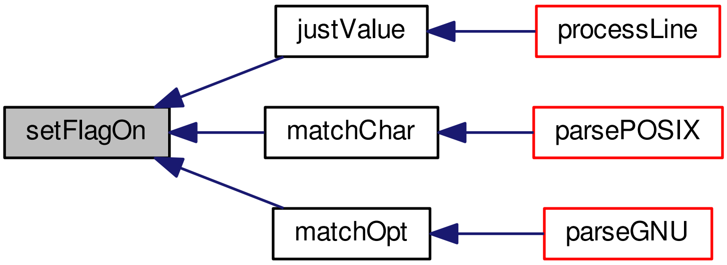
Referenced by justValue(), matchChar(), and matchOpt().

```
00860 {
00861   if( !valueStoreOK() )
00862     return false;
00863         for( int i = 0 ; i < option_counter ; i++ ){
00864                 if( strcmp( options[i], option ) == 0 ){
00865                         values[ optionindex[i] ] = (char*) malloc((strlen(
      TRUE_FLAG)+1)*sizeof(char));
00866                         strcpy( values[ optionindex[i] ]  ,  TRUE_FLAG );
00867       return true;
00868     }
00869         }
00870         return false;
00871 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

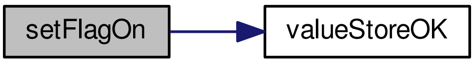


4.1.3.71 bool setFlagOn (char *optchar*) `[private]`

Definition at line 889 of file anyoption.cpp.

```
{
  if( !valueStoreOK() )
    return false;
        for( int i = 0 ; i < optchar_counter ; i++ ){
                if( optionchars[i] == option ){
                        values[ optcharindex[i] ] = (char*) malloc((strlen(TRUE_FLAG)+1)*sizeof(char));
      strcpy( values[ optcharindex[i] ] , TRUE_FLAG );
      return true;
    }
        }
        return false;
}
```

Here is the call graph for this function:



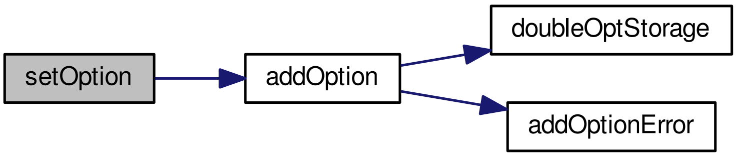
4.1.3.72 void setOption (const char * *opt_string*)

Definition at line 477 of file anyoption.cpp.

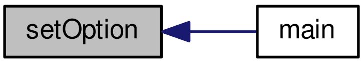
Referenced by main().

```
{
  addOption( opt , COMMON_OPT );
  g_value_counter++;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

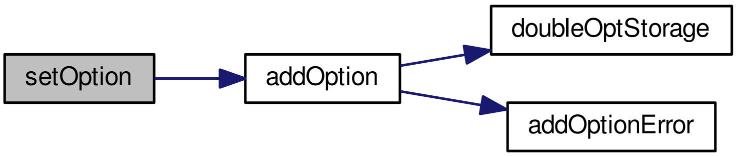


4.1.3.73 void setOption (char *opt_char*)

Definition at line 484 of file anyoption.cpp.

```
00485 {
00486   addOption( opt , COMMON_OPT );
00487   g_value_counter++;
00488 }
```

Here is the call graph for this function:

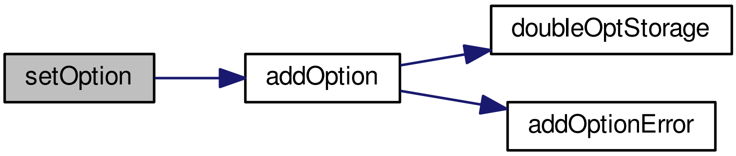


4.1.3.74 void setOption (const char ∗ *opt_string,* char *opt_char*)

Definition at line 491 of file anyoption.cpp.

```
{
  addOption( opt , COMMON_OPT );
  addOption( optchar , COMMON_OPT );
  g_value_counter++;
}
```

Here is the call graph for this function:



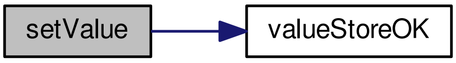
4.1.3.75 bool setValue (const char ∗ *option,* char ∗ *value*) [private]

Definition at line 844 of file anyoption.cpp.

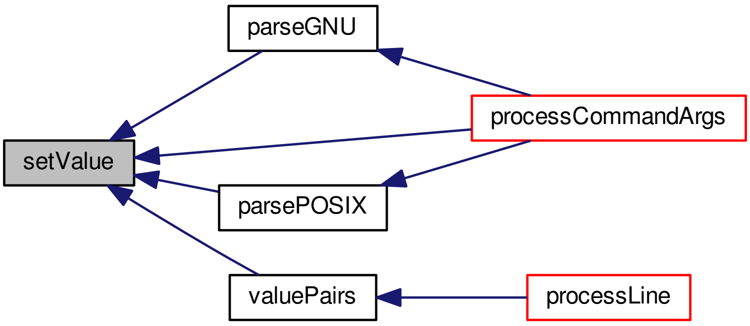
Referenced by parseGNU(), parsePOSIX(), processCommandArgs(), and valuePairs().

```
{
  if( !valueStoreOK() )
    return false;
        for( int i = 0 ; i < option_counter ; i++ ){
                if( strcmp( options[i], option ) == 0 ){
                        values[ optionindex[i] ] = (char*) malloc((strlen(value)+1)*sizeof
(char));
                        strcpy( values[ optionindex[i] ], value );
      return true;
    }
        }
        return false;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

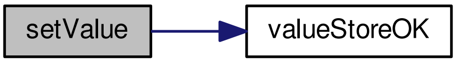


4.1.3.76 bool setValue (char *optchar,* char ∗ *value*) [private]

Definition at line 874 of file anyoption.cpp.

```
{
  if( !valueStoreOK() )
    return false;
        for( int i = 0 ; i < optchar_counter ; i++ ){
                if( optionchars[i] == option ){
                        values[ optcharindex[i] ] = (char*) malloc((strlen(value)+1)*sizeof(char));
                        strcpy( values[ optcharindex[i] ],  value );
      return true;
    }
        }
        return false;
}
```

Here is the call graph for this function:



4.1.3.77 void setVerbose ( )

Definition at line 323 of file anyoption.cpp.

```
{
  verbose = true ;
}
```

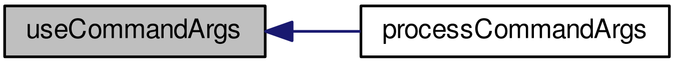
4.1.3.78 void useCommandArgs (int *_argc,* char ∗∗ *_argv*)

Definition at line 368 of file anyoption.cpp.

Referenced by processCommandArgs().

```
{
  argc = _argc;
  argv = _argv;
  command_set = true;
  appname = argv[0];
  if(argc > 1) hasoptions = true;
}
```

Here is the caller graph for this function:



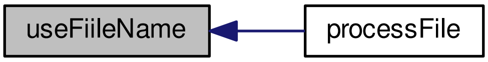
4.1.3.79 void useFiileName (const char ∗ *_filename*)

Definition at line 378 of file anyoption.cpp.

Referenced by processFile().

```
{
  filename = _filename;
  file_set = true;
}
```

Here is the caller graph for this function:



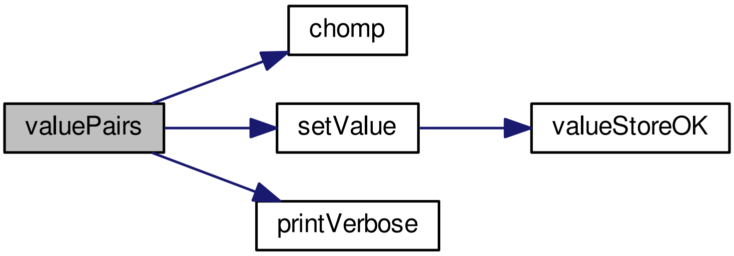
4.1.3.80 void valuePairs (char ∗ *type,* char ∗ *value*) [private]

Definition at line 1065 of file anyoption.cpp.

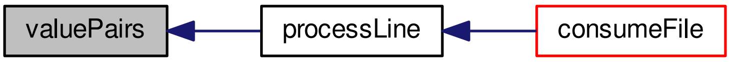
Referenced by processLine().

```
{
  if ( strlen(chomp(type)) == 1  ){ /* this is a char option */
    for( int i = 0 ; i < optchar_counter ; i++ ){
      if(  optionchars[i] == type[0]  ){ /* match */
        if( optchartype[i] == COMMON_OPT ||
            optchartype[i] == FILE_OPT )
        {
           setValue( type[0] , chomp(value) );
           return;
        }
      }
    }
  }
  /* if no char options matched */
  for( int i = 0 ; i < option_counter ; i++ ){
    if( strcmp( options[i], type ) == 0 ){ /* match */
      if( optiontype[i] == COMMON_OPT ||
          optiontype[i] == FILE_OPT )
      {
         setValue( type , chomp(value) );
         return;
      }
    }
  }
        printVerbose( "Unknown option in resourcefile : " );
  printVerbose( type );
  printVerbose( );
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



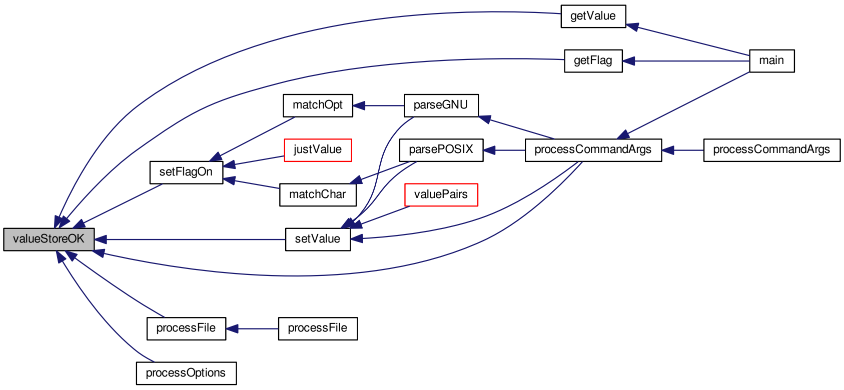
4.1.3.81 bool valueStoreOK ( ) `[private]`

Definition at line 761 of file anyoption.cpp.

Referenced by getFlag(), getValue(), processCommandArgs(), processFile(), processOptions(), setFlagOn(), and setValue().

```
{
  int size= 0;
  if( !set ){
    if( g_value_counter > 0 ){
      size = g_value_counter * sizeof(char*);
      values = (char**)malloc( size );
      for( int i = 0 ; i < g_value_counter ; i++)
        values[i] = NULL;
      set = true;
    }
  }
  return  set;
}
```

Here is the caller graph for this function:



4.1.4 Member Data Documentation

4.1.4.1 char∗ appname `[private]`

Definition at line 166 of file anyoption.h.

Referenced by init(), and useCommandArgs().

4.1.4.2 int argc `[private]`

Definition at line 163 of file anyoption.h.

Referenced by init(), processCommandArgs(), and useCommandArgs().

4.1.4.3 char∗∗ argv `[private]`

Definition at line 164 of file anyoption.h.

Referenced by getArgv(), init(), processCommandArgs(), and useCommandArgs().

4.1.4.4 bool autousage [private]

Definition at line 219 of file anyoption.h.

Referenced by autoUsagePrint(), init(), and printAutoUsage().

4.1.4.5 bool command_set [private]

Definition at line 196 of file anyoption.h.

Referenced by CommandSet(), init(), and useCommandArgs().

4.1.4.6 char comment [private]

Definition at line 209 of file anyoption.h.

Referenced by consumeFile(), and init().

4.1.4.7 char delimiter [private]

Definition at line 210 of file anyoption.h.

Referenced by init(), and processLine().

4.1.4.8 char endofline [private]

Definition at line 211 of file anyoption.h.

Referenced by consumeFile(), and init().

4.1.4.9 char equalsign [private]

Definition at line 208 of file anyoption.h.

Referenced by init(), parseGNU(), and parsePOSIX().

4.1.4.10 char file_comment_char [private]

Definition at line 207 of file anyoption.h.

Referenced by init(), and setFileDelimiterChar().

4.1.4.11 char file_delimiter_char [private]

Definition at line 206 of file anyoption.h.

Referenced by init(), and setFileCommentChar().

4.1.4.12 bool file_set [private]

Definition at line 197 of file anyoption.h.

Referenced by FileSet(), init(), and useFiileName().

4.1.4.13 const char∗ filename [private]

Definition at line 165 of file anyoption.h.

Referenced by init(), readFile(), and useFiileName().

4.1.4.14 int g_value_counter [private]

Definition at line 189 of file anyoption.h.

Referenced by addOption(), init(), setCommandFlag(), setCommandOption(), setFileFlag(), setFileOption(), set←Flag(), setOption(), and valueStoreOK().

4.1.4.15 bool hasoptions [private]

Definition at line 218 of file anyoption.h.

Referenced by hasOptions(), init(), and useCommandArgs().

4.1.4.16 char long_opt_prefix[MAX_LONG_PREFIX_LENGTH+1] [private]

Definition at line 205 of file anyoption.h.

Referenced by init(), processCommandArgs(), and setCommandLongPrefix().

4.1.4.17 int max_char_options [private]

Definition at line 181 of file anyoption.h.

Referenced by addOption(), alloc(), doubleCharStorage(), and init().

4.1.4.18 int max_legal_args [private]

Definition at line 170 of file anyoption.h.

Referenced by init(), and processCommandArgs().

4.1.4.19 int max_options [private]

Definition at line 174 of file anyoption.h.

Referenced by addOption(), alloc(), doubleOptStorage(), and init().

4.1.4.20 int max_usage_lines [private]

Definition at line 193 of file anyoption.h.

Referenced by addUsage(), alloc(), doubleUsageStorage(), and init().

4.1.4.21 bool mem_allocated [private]

Definition at line 198 of file anyoption.h.

Referenced by alloc(), init(), and ∼AnyOption().

4.1.4.22 int new_argc [private]

Definition at line 169 of file anyoption.h.

Referenced by getArgc(), getArgv(), init(), and processCommandArgs().

4.1.4.23 int∗ new_argv [private]

Definition at line 168 of file anyoption.h.

Referenced by cleanup(), getArgv(), init(), and processCommandArgs().

4.1.4.24 char nullterminate [private]

Definition at line 213 of file anyoption.h.

Referenced by chomp(), init(), and processLine().

4.1.4.25 bool once [private]

Definition at line 216 of file anyoption.h.

Referenced by init(), and printUsage().

4.1.4.26 char opt_prefix_char [private]

Definition at line 204 of file anyoption.h.

Referenced by init(), processCommandArgs(), and setCommandPrefixChar().

4.1.4.27 int optchar_counter [private]

Definition at line 185 of file anyoption.h.

Referenced by addOption(), getFlag(), getValue(), init(), justValue(), matchChar(), setFlagOn(), setValue(), and valuePairs().

4.1.4.28 int∗ optcharindex [private]

Definition at line 184 of file anyoption.h.

Referenced by addOption(), alloc(), cleanup(), doubleCharStorage(), getFlag(), getValue(), setFlagOn(), and set←↩ Value().

4.1.4.29 int∗ optchartype [private]

Definition at line 183 of file anyoption.h.

Referenced by addOption(), alloc(), cleanup(), doubleCharStorage(), justValue(), matchChar(), and valuePairs().

4.1.4.30 int option_counter [private]

Definition at line 178 of file anyoption.h.

Referenced by addOption(), getFlag(), getValue(), init(), justValue(), matchOpt(), setFlagOn(), setValue(), and valuePairs().

4.1.4.31 char* optionchars [private]

Definition at line 182 of file anyoption.h.

Referenced by addOption(), alloc(), cleanup(), doubleCharStorage(), getFlag(), getValue(), justValue(), match↩Char(), setFlagOn(), setValue(), and valuePairs().

4.1.4.32 int* optionindex [private]

Definition at line 177 of file anyoption.h.

Referenced by addOption(), alloc(), cleanup(), doubleOptStorage(), getFlag(), getValue(), setFlagOn(), and set↩Value().

4.1.4.33 const char** options [private]

Definition at line 175 of file anyoption.h.

Referenced by addOption(), alloc(), cleanup(), doubleOptStorage(), getFlag(), getValue(), justValue(), matchOpt(), parseGNU(), processCommandArgs(), setFlagOn(), setValue(), and valuePairs().

4.1.4.34 int* optiontype [private]

Definition at line 176 of file anyoption.h.

Referenced by addOption(), alloc(), cleanup(), doubleOptStorage(), justValue(), matchOpt(), and valuePairs().

4.1.4.35 bool posix_style [private]

Definition at line 199 of file anyoption.h.

Referenced by init(), noPOSIX(), and POSIX().

4.1.4.36 bool print_help [private]

Definition at line 202 of file anyoption.h.

4.1.4.37 bool print_usage [private]

Definition at line 201 of file anyoption.h.

4.1.4.38 bool set [private]

Definition at line 215 of file anyoption.h.

4.1.4.39 const char** usage [private]

Definition at line 192 of file anyoption.h.

Referenced by addUsage(), alloc(), cleanup(), doubleUsageStorage(), and printUsage().

4.1.4.40 int usage_lines [private]

Definition at line 194 of file anyoption.h.

Referenced by addUsage(), init(), and printUsage().

4.1.4.41 char** values [private]

Definition at line 188 of file anyoption.h.

Referenced by cleanup(), getFlag(), getValue(), init(), setFlagOn(), setValue(), and valueStoreOK().

4.1.4.42 bool verbose [private]

Definition at line 200 of file anyoption.h.

Referenced by init(), printVerbose(), and setVerbose().

4.1.4.43 char whitespace [private]

Definition at line 212 of file anyoption.h.

Referenced by chomp(), init(), and parsePOSIX().

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

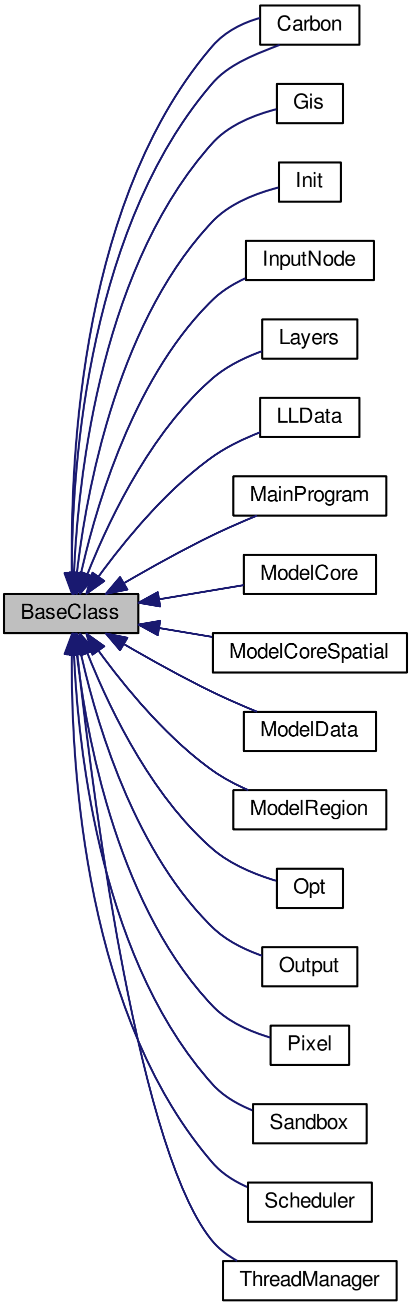
- /home/lobianco/git/ffsm_pp/src/anyoption.h
- /home/lobianco/git/ffsm_pp/src/anyoption.cpp

4.2 BaseClass Class Reference

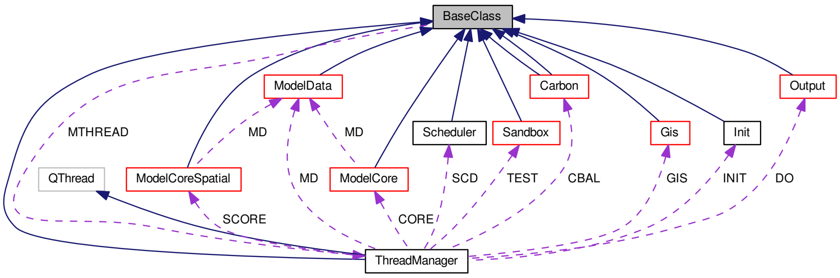
Base class for the regmas application.

`#include <BaseClass.h>`

Inheritance diagram for BaseClass:



Collaboration diagram for BaseClass:



Public Member Functions

- BaseClass ()
- ∼BaseClass ()
- void msgOut (const int &msgCode_h, const string &msg_h, const bool &refreshGUI_h=true) const

  Overloaded function to print the output log.
- void msgOut (const int &msgCode_h, const int &msg_h, const bool &refreshGUI_h=true) const

  Overloaded function to print the output log.
- void msgOut (const int &msgCode_h, const double &msg_h, const bool &refreshGUI_h=true) const

  Overloaded function to print the output log.
- int s2i (const string &string_h) const

  string to integer conversion
- double s2d (const string &string_h) const

  string to double conversion
- double s2d (const string &string_h, const bool &replaceComma) const

  string to double conversion
- bool s2b (const string &string_h) const

  string to bool conversion
- string i2s (const int &int_h) const

  integer to string conversion
- string d2s (const double &double_h) const

  double to string conversion
- string b2s (const bool &bool_h) const

  bool to string conversion
- vector< int > s2i (const vector< string > &string_h) const

  string to integer conversion (vector)
- vector< double > s2d (const vector< string > &string_h, const bool &replaceComma=false) const

  string to double conversion (vector)
- vector< bool > s2b (const vector< string > &string_h) const

  string to bool conversion (vector)
- vector< string > i2s (const vector< int > &int_h) const

  integer to string conversion (vector)
- vector< string > d2s (const vector< double > &double_h) const

  double to string conversion (vector)
- vector< string > b2s (const vector< bool > &bool_h) const

  bool to string conversion (vector)
- int getType (const string &type_h) const

Return a type according to enum TYPE_∗ from a string (eg: "string" -> TYPE_STRING (2))
- void refreshGUI () const

  Ping to periodically return the control to the GUI.
- template< typename T >
  string toString (const T &x) const
- template< typename T >
  T stringTo (const std::string &s) const
- int vSum (const vector< int > &vector_h) const
- double vSum (const vector< double > &vector_h) const
- int vSum (const vector< vector< int > > &vector_h) const
- double vSum (const vector< vector< double > > &vector_h) const
- void tokenize (const string &str, vector< string > &tokens, const string &delimiter=" ") const

  Tokenize a string using a delimiter (default is space)
- void untokenize (string &str, vector< string > &tokens, const string &delimiter=" ") const
- template< typename K , typename V >
  V findMap (const map< K, V > &mymap, const K &key, const int &error_level=MSG_CRITICAL_ERROR, const V &notFoundValue=numeric_limits< V >::min()) const

  Lookup a map for a value. Return the value starting from the key.
- template< typename K , typename V >
  void changeMapValue (map< K, V > &mymap, const K &key, const V &value, const int &error_level=MSG_CRITICAL_ERROR)

  Change the value stored in a map given the key and the new value.
- template< typename K , typename V >
  void incrMapValue (map< K, V > &mymap, const K &key, const V &value, const int &error_level=MSG_CRITICAL_ERROR)

  Increments a value stored in a map of the specified value, given the key.
- template< typename K , typename V >
  void incrOrAddMapValue (map< K, V > &mymap, const K &key, const V &value)

  Increments a value stored in a map of the specified value, given the key.
- template< typename K , typename V >
  void resetMapValues (map< K, V > &mymap, const V &value)

  Reset all values stored in a map to the specified one.
- template< typename K , typename V >
  map< K, V > vectorToMap (const vector< K > &keys, const V &value=0.0)

  Returns a map built using the given vector and the given (scalar) value as keys/values pairs.
- template< typename T >
  vector< T > positionsToContent (const vector< T > &vector_h, const vector< int > &positions)

  Return a vector of content from a vector and a vector of positions (int)
- template< typename V >
  void debugMap (const map< iisskey, V > &mymap)

  Debug a map.
- template< typename K , typename V >
  void debugMap (const map< K, V > &mymap, const K &key)
- template< typename K >
  int getMaxPos (const vector< K > &v)

  Returns the position of the maximum element in the vector (the last one in case of multiple equivalent maxima)
- template< typename K >
  int getMinPos (const vector< K > &v)

  Returns the position of the minimum element in the vector (the first one in case of multiple equivalent minima)
- template< typename K >
  K getMax (const vector< K > &v)

  Returns the value of the maximum element in the vector (the last one in case of multiple equivalent maxima)
- template< typename K >
  K getMin (const vector< K > &v)

Returns the value of the minimum element in the vector (the first one in case of multiple equivalent minima)

- template<typename K >
  double getAvg (const vector< K > &v)

  Returns the average of the elements in the vector.
- template<typename K >
  double getSd (const vector< K > &v, bool sample=true)
- template<typename K >
  int getPos (const K &element, const vector< K > &v, const int &msgCode_h=MSG_CRITICAL_ERROR)
- template<typename K >
  bool inVector (const K &element, const vector< K > &v)
- double normSample (const double &avg, const double &stdev, const double &minval=NULL, const double &maxval=NULL) const

  Sample from a normal distribution with bounds. Slower (double time, but still you see the diff only after milion of loops).
- template<typename K >
  K normSample (normal_distribution< K > &d, std::mt19937 &gen, const K &minval=NULL, const K &maxval=NULL) const

  Sample from a normal distribution with bounds. Faster (half time) as the normal_distribution is made only once.
- template<typename T >
  std::string toString (const T &x) const

Protected Attributes

- ThreadManager ∗ MTHREAD

  Pointer to the Thread manager.

Private Member Functions

- void msgOut2 (const int &msgCode_h, const string &msg_h, const bool &refreshGUI_h) const

  Do the job of the overloaded functions.

4.2.1 Detailed Description

Base class for the regmas application.

This class is the base class for all classes in regmas. \ It provides common methods in all parts of the application for printing the output, converting strings vs. values or regularly "ping" the GUI. \

Author

Antonello Lobianco

Definition at line 236 of file BaseClass.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 BaseClass ( )

Definition at line 31 of file BaseClass.cpp.

```
00032 {
00033   MTHREAD=NULL;
00034 }
```

4.2.2.2 ∼BaseClass ( )

Definition at line 36 of file BaseClass.cpp.

```
00037 {
00038 
00039 }
```

4.2.3 Member Function Documentation

4.2.3.1 string b2s (const bool & *bool_h*) const

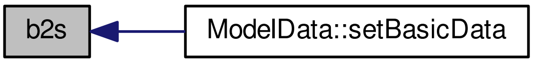
bool to string conversion

Definition at line 234 of file BaseClass.cpp.

Referenced by ModelData::setBasicData().

```
00234                                     {
00235   if (bool_h) return "true";
00236   else return "false";
00237 }
```

Here is the caller graph for this function:



4.2.3.2 vector< string > b2s (const vector< bool > & *bool_h*) const

bool to string conversion (vector)

Definition at line 294 of file BaseClass.cpp.

```
00294                                                     {
00295   vector <string> valuesAsString;
00296   for (uint i=0;i<bool_h.size();i++){
00297     if(bool_h[i]) valuesAsString.push_back("true");
00298     else valuesAsString.push_back("false");
00299   }
00300   return valuesAsString;
00301 }
```

4.2.3.3 void changeMapValue (map< K, V > & *mymap*, const K & *key*, const V & *value*, const int & *error_level* = MSG_CRITICAL_ERROR) [inline]

Change the value stored in a map given the key and the new value.

Definition at line 296 of file BaseClass.h.

```
00296
                                                  {
00297       typename map<K, V>::iterator p;
00298       p=mymap.find(key);
00299       if(p != mymap.end()) {
00300           p->second = value;
00301           return;
00302       }
00303       else {
00304           msgOut(error_level, "Error in finding a value in a map (no value found)");
00305       }
00306   }
```

4.2.3.4 string d2s (const double & *double_h*) const

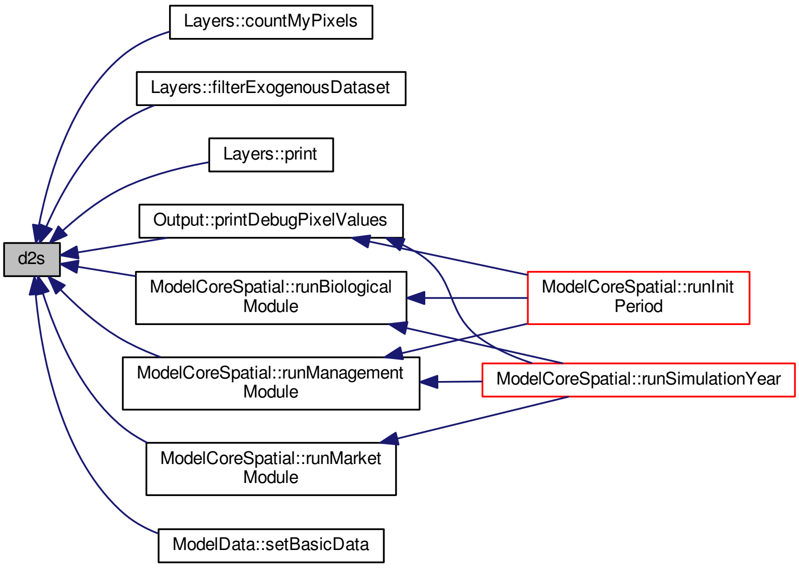
double to string conversion

Definition at line 224 of file BaseClass.cpp.

Referenced by Layers::countMyPixels(), Layers::filterExogenousDataset(), Layers::print(), Output::printDebugPixelValues(), ModelCoreSpatial::runBiologicalModule(), ModelCoreSpatial::runManagementModule(), ModelCoreSpatial::runMarketModule(), and ModelData::setBasicData().

```
00224                                                      {
00225   //ostringstream out;
00226   //out<<double_h;
00227   //return out.str();
00228   char  outChar[24];
00229   snprintf ( outChar, sizeof(outChar), "%f", double_h );
00230   return string(outChar);
00231 }
```

Here is the caller graph for this function:


Layers::countMyPixels
Layers::filterExogenousDataset
Layers::print
Output::printDebugPixelValues
d2s
ModelCoreSpatial::runBiological Module
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::runSimulationYear
ModelCoreSpatial::runMarket Module
ModelData::setBasicData

4.2.3.5 vector< string > d2s (const vector< double > & *double_h*) const

double to string conversion (vector)

Definition at line 285 of file BaseClass.cpp.

```
00285                                                     {
00286   vector <string> valuesAsString;
00287   for (uint i=0;i<double_h.size();i++){
00288         valuesAsString.push_back(d2s(double_h[i]));
00289   }
00290   return valuesAsString;
00291 }
```

4.2.3.6 void debugMap (const map< iisskey, V > & *mymap*) [inline]

Debug a map.

Definition at line 365 of file BaseClass.h.

```
00365                                                                     {
00366     iisskey mykey(NULL,NULL,"","");
00367     debugMap(mymap, mykey);
00368   }
```

4.2.3.7 void debugMap (const map< K, V > & *mymap*, const K & *key*) [inline]

Definition at line 369 of file BaseClass.h.

```
00369                                                                                         {
00370     cout<<"Debugging a map" << endl;
00371     for (auto const &themap: mymap) {
00372       if(themap.first.filter(key)){
00373        cout << themap.first.print() <<  '\t' << themap.second << endl;
00374       }
00375     }
00376   }
```

4.2.3.8 V findMap (const map< K, V > & *mymap*, const K & *key*, const int & *error_level* = MSG_CRITICAL_ERROR, const V & *notFoundValue* = numeric_limits<V>::min()) const [inline]

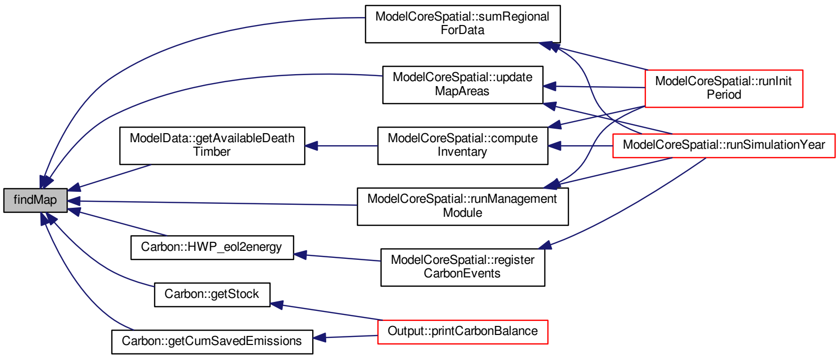
Lookup a map for a value. Return the value starting from the key.

Definition at line 283 of file BaseClass.h.

Referenced by ModelData::getAvailableDeathTimber(), Carbon::getCumSavedEmissions(), Carbon::getStock(), Carbon::HWP_eol2energy(), ModelCoreSpatial::runManagementModule(), ModelCoreSpatial::sumRegionalForData(), and ModelCoreSpatial::updateMapAreas().

```
00283
                                                                         {
00284     typename map<K, V>::const_iterator p;
00285     p=mymap.find(key);
00286     if(p != mymap.end()) {
00287       return p->second;
00288     }
00289     else {
00290       msgOut(error_level, "Error in finding a value in a map (no value found)");
00291       return notFoundValue;
00292     }
00293   }
```

Here is the caller graph for this function:


ModelCoreSpatial::sumRegional ForData
ModelCoreSpatial::update MapAreas
ModelCoreSpatial::runInit Period
ModelData::getAvailableDeath Timber
ModelCoreSpatial::compute Inventary
ModelCoreSpatial::runSimulationYear
findMap
ModelCoreSpatial::runManagement Module
Carbon::HWP_eol2energy
ModelCoreSpatial::register CarbonEvents
Carbon::getStock
Output::printCarbonBalance
Carbon::getCumSavedEmissions

4.2.3.9 double getAvg (const vector< K > & *v*) [inline]

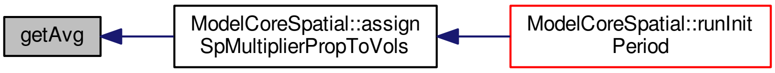
Returns the average of the elements in the vector.

Definition at line 396 of file BaseClass.h.

Referenced by ModelCoreSpatial::assignSpMultiplierPropToVols().

```
00396                                                             {
00397     return v.size()==0 ? 0.0 : vSum(v)/ ( (double) v.size() );
00398   }
```

Here is the caller graph for this function:


getAvg
ModelCoreSpatial::assign SpMultiplierPropToVols
ModelCoreSpatial::runInit Period

4.2.3.10 K getMax (const vector< K > & *v*) [inline]

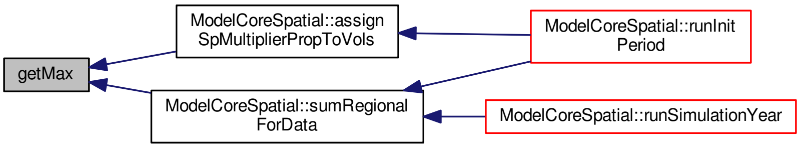
Returns the value of the maximum element in the vector (the last one in case of multiple equivalent maxima)

Definition at line 388 of file BaseClass.h.

Referenced by ModelCoreSpatial::assignSpMultiplierPropToVols(), and ModelCoreSpatial::sumRegionalForData().

```
00388                                                      {
00389     return *minmax_element(v.begin(), v.end()).second;
00390   }
```

Here is the caller graph for this function:


getMax
ModelCoreSpatial::assign SpMultiplierPropToVols
ModelCoreSpatial::sumRegional ForData
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

4.2.3.11 int getMaxPos (const vector< K > & *v*) `[inline]`

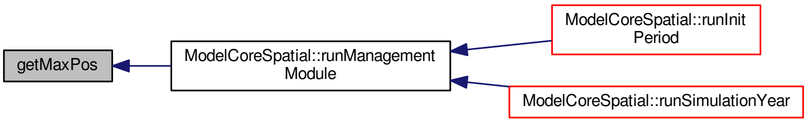
Returns the position of the maximum element in the vector (the last one in case of multiple equivalent maxima)

Definition at line 380 of file BaseClass.h.

Referenced by ModelCoreSpatial::runManagementModule().

```
00380                                                          {
00381     return (minmax_element(v.begin(), v.end()).second - v.begin());
00382   }
```

Here is the caller graph for this function:


getMaxPos
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

4.2.3.12 K getMin (const vector< K > & *v*) `[inline]`

Returns the value of the minimum element in the vector (the first one in case of multiple equivalent minima)

Definition at line 392 of file BaseClass.h.

```
00392                                                     {
00393     return *minmax_element(v.begin(), v.end()).first;
00394   }
```

4.2.3.13 int getMinPos (const vector< K > & *v*) `[inline]`

Returns the position of the minimum element in the vector (the first one in case of multiple equivalent minima)

Definition at line 384 of file BaseClass.h.

```
00384                                                          {
00385     return (minmax_element(v.begin(), v.end()).first - v.begin());
00386   }
```

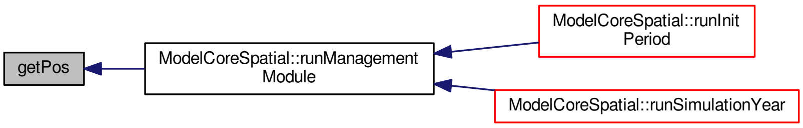
4.2.3.14 int getPos (const K & *element*, const vector< K > & *v*, const int & *msgCode_h* = MSG_CRITICAL_ERROR) `[inline]`

Definition at line 418 of file BaseClass.h.

Referenced by ModelCoreSpatial::runManagementModule().

```
00418
                {
00419     for(unsigned int i=0; i<v.size(); i++){
00420       if(v[i]== element) return i;
00421     }
00422     msgOut(msgCode_h, "Element not found in vector in getPos()");
00423     return -1;
00424   }
```

Here is the caller graph for this function:


getPos
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

4.2.3.15 double getSd (const vector< K > & *v*, bool *sample* = `true`) `[inline]`

Returns the sd of the elements of a vector. Default to sample sd.

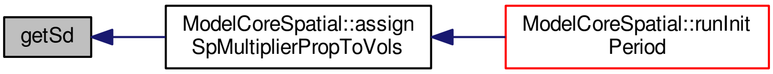
See `http://stackoverflow.com/questions/7616511/calculate-mean-and-standard-deviation-from` where there is also an example for covariance

Definition at line 405 of file BaseClass.h.

Referenced by ModelCoreSpatial::assignSpMultiplierPropToVols().

```
00405                                                                         {
00406     if (v.size()==0) return 0.0;
00407     int sampleCorrection = sample==true?1:0;
00408     double sum = std::accumulate(std::begin(v), std::end(v), 0.0);
00409     double m =  sum / v.size();
00410     double accum = 0.0;
00411     std::for_each (std::begin(v), std::end(v), [&](const double d) {
00412       accum += (d - m) * (d - m);
00413     });
00414     double stdev = sqrt(accum / ( (double) (v.size()-sampleCorrection)));
00415     return stdev;
00416   }
```

Here is the caller graph for this function:


getSd
ModelCoreSpatial::assign SpMultiplierPropToVols
ModelCoreSpatial::runInit Period

4.2.3.16 int getType (const string & *type_h*) const

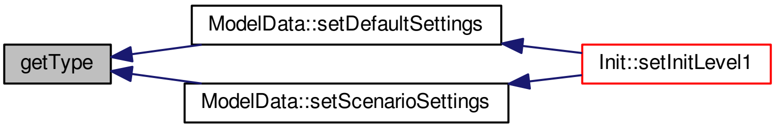
Return a type according to enum TYPE_∗ from a string (eg: "string" -> TYPE_STRING (2))

Definition at line 305 of file BaseClass.cpp.

Referenced by ModelData::setDefaultSettings(), and ModelData::setScenarioSettings().

```
00305                                            {
00306   int toReturn=0;
00307   if (type_h == "int")          toReturn = TYPE_INT;
00308   else if (type_h == "double") toReturn = TYPE_DOUBLE;
00309   else if (type_h == "string") toReturn = TYPE_STRING;
00310   else if (type_h == "bool")    toReturn = TYPE_BOOL;
00311   else msgOut(MSG_CRITICAL_ERROR, "Unknow type "+type_h+".");
00312   return toReturn;
00313 }
```

Here is the caller graph for this function:



4.2.3.17 string i2s (const int & *int_h*) const

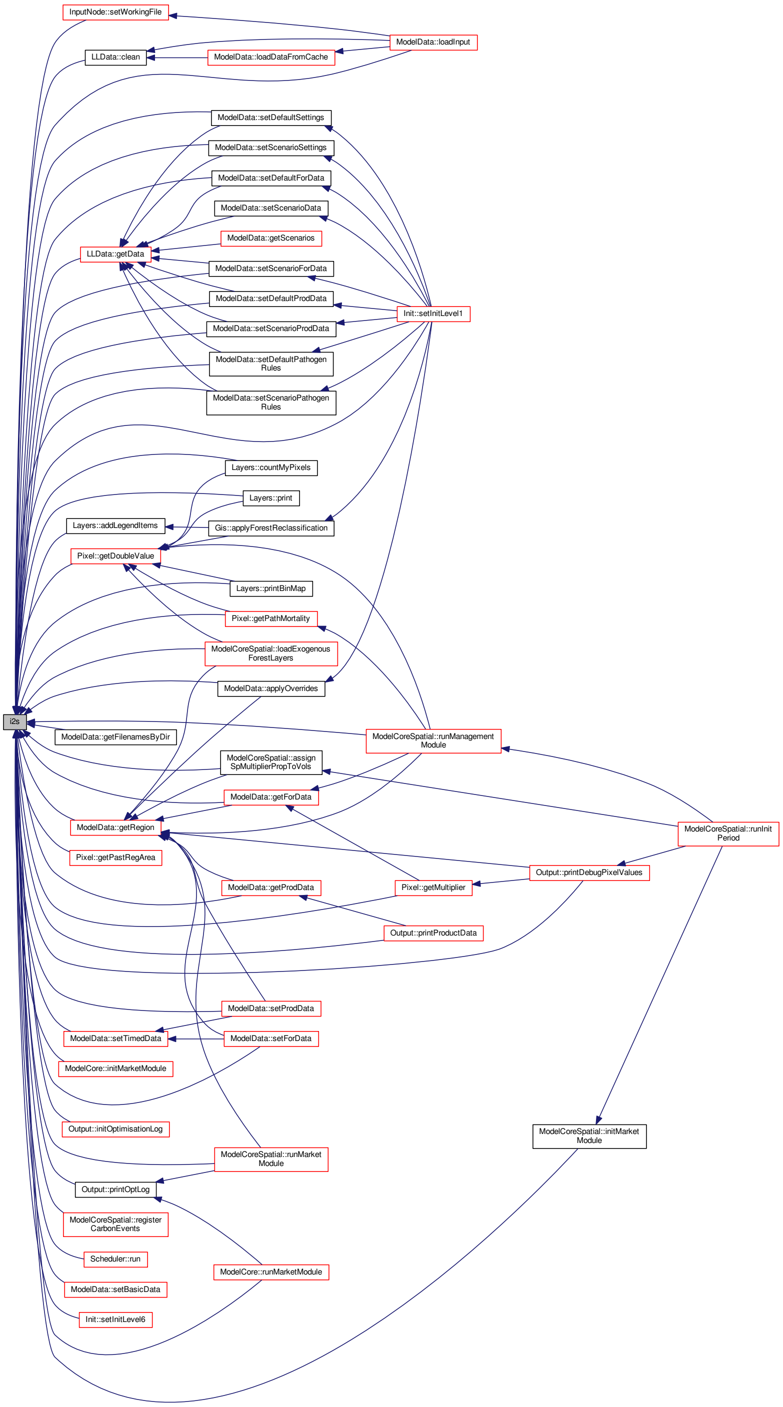
integer to string conversion

Definition at line 214 of file BaseClass.cpp.

Referenced by Layers::addLegendItems(), ModelData::applyOverrides(), ModelCoreSpatial::assignSpMultiplierPropToVols(), LLData::clean(), Layers::countMyPixels(), LLData::getData(), Pixel::getDoubleValue(), ModelData::getFilenamesByDir(), ModelData::getForData(), Pixel::getMultiplier(), Pixel::getPastRegArea(), Pixel::getPathMortality(), ModelData::getProdData(), ModelData::getRegion(), ModelCore::initMarketModule(), ModelCoreSpatial::initMarketModule(), Output::initOptimisationLog(), ModelCoreSpatial::loadExogenousForestLayers(), ModelData::loadInput(), Layers::print(), Layers::printBinMap(), Output::printDebugPixelValues(), Output::printOptLog(), Output::printProductData(), ModelCoreSpatial::registerCarbonEvents(), Scheduler::run(), ModelCoreSpatial::runManagementModule(), ModelCore::runMarketModule(), ModelCoreSpatial::runMarketModule(), ModelData::setBasicData(), ModelData::setDefaultForData(), ModelData::setDefaultPathogenRules(), ModelData::setDefaultProdData(), ModelData::setDefaultSettings(), ModelData::setForData(), Init::setInitLevel1(), Init::setInitLevel6(), ModelData::setProdData(), ModelData::setScenarioForData(), ModelData::setScenarioPathogenRules(), ModelData::setScenarioProdData(), ModelData::setScenarioSettings(), ModelData::setTimedData(), and InputNode::setWorkingFile().

```
00214                                    {
00215   //ostringstream out;
00216   //out<<int_h;
00217   //return out.str();
00218   char  outChar[24];
00219   snprintf ( outChar, sizeof(outChar), "%d", int_h );
00220   return string(outChar);
00221 }
```

Here is the caller graph for this function:


InputNode::setWorkingFile
ModelData::loadInput
LLData::clean
ModelData::loadDataFromCache
ModelData::setDefaultSettings
ModelData::setScenarioSettings
ModelData::setDefaultForData
ModelData::setScenarioData
ModelData::getScenarios
LLData::getData
ModelData::setScenarioForData
ModelData::setDefaultProdData
Init::setInitLevel1
ModelData::setScenarioProdData
ModelData::setDefaultPathogen Rules
ModelData::setScenarioPathogen Rules
Layers::countMyPixels
Layers::print
Layers::addLegendItems
Gis::applyForestReclassification
Pixel::getDoubleValue
Layers::printBinMap
Pixel::getPathMortality
ModelCoreSpatial::loadExogenous ForestLayers
ModelData::applyOverrides
i2s
ModelData::getFilenamesByDir
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::assign SpMultiplierPropToVols
ModelData::getForData
ModelData::getRegion
ModelCoreSpatial::runInit Period
Pixel::getPastRegArea
Output::printDebugPixelValues
ModelData::getProdData
Pixel::getMultiplier
Output::printProductData
ModelData::setProdData
ModelData::setTimedData
ModelData::setForData
ModelCore::initMarketModule
Output::initOptimisationLog
ModelCoreSpatial::initMarket Module
ModelCoreSpatial::runMarket Module
Output::printOptLog
ModelCoreSpatial::register CarbonEvents
Scheduler::run
ModelCore::runMarketModule
ModelData::setBasicData
Init::setInitLevel6

4.2.3.18 vector< string > i2s (const vector< int > & *int_h*) const

integer to string conversion (vector)

Definition at line 276 of file BaseClass.cpp.

```
00276                                                       {
00277   vector <string> valuesAsString;
00278   for (uint i=0;i<int_h.size();i++){
00279         valuesAsString.push_back(i2s(int_h[i]));
00280   }
00281   return valuesAsString;
00282 }
```

4.2.3.19 void incrMapValue (map< K, V > & *mymap*, const K & *key*, const V & *value*, const int & *error_level* = MSG_CRITICAL_ERROR) [inline]

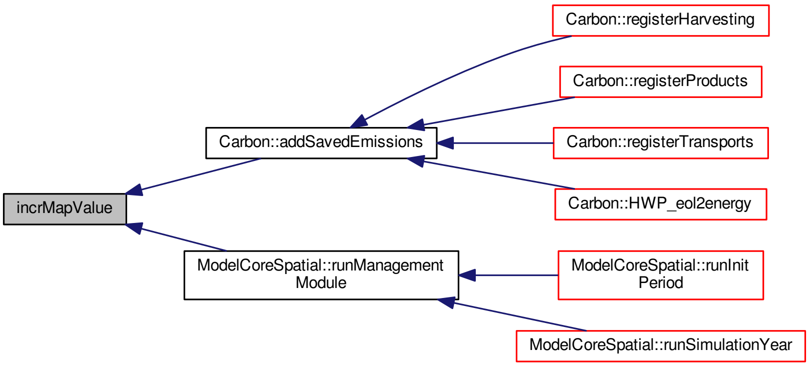
Increments a value stored in a map of the specified value, given the key.

Definition at line 309 of file BaseClass.h.

Referenced by Carbon::addSavedEmissions(), and ModelCoreSpatial::runManagementModule().

```
00309
                                                    {
00310       typename map<K, V>::iterator p;
00311       p=mymap.find(key);
00312       if(p != mymap.end()) {
00313           p->second = p->second + value;
00314           return;
00315       }
00316       else {
00317           msgOut(error_level, "Error in finding a value in a map (no value found)");
00318       }
00319   }
```

Here is the caller graph for this function:


Carbon::registerHarvesting
Carbon::registerProducts
Carbon::addSavedEmissions
Carbon::registerTransports
incrMapValue
Carbon::HWP_eol2energy
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

4.2.3.20 void incrOrAddMapValue (map< K, V > & *mymap*, const K & *key*, const V & *value*) [inline]

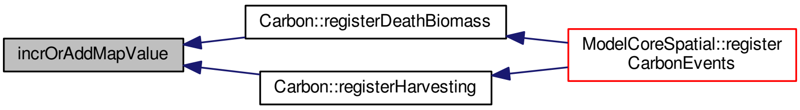
Increments a value stored in a map of the specified value, given the key.

Definition at line 322 of file BaseClass.h.

Referenced by Carbon::registerDeathBiomass(), and Carbon::registerHarvesting().

```
00322
      {
00323       typename map<K, V>::iterator p;
00324       p=mymap.find(key);
00325       if(p != mymap.end()) {
00326           // We found the key, we gonna add the value..
00327           p->second = p->second + value;
00328           return;
00329       }
00330       else {
00331           // We didn't find the key, we gonna add it together with the value
00332           pair<K,V> myPair(key,value);
00333           mymap.insert(myPair);
00334       }
00335   }
```

Here is the caller graph for this function:


incrOrAddMapValue
Carbon::registerDeathBiomass
Carbon::registerHarvesting
ModelCoreSpatial::register CarbonEvents

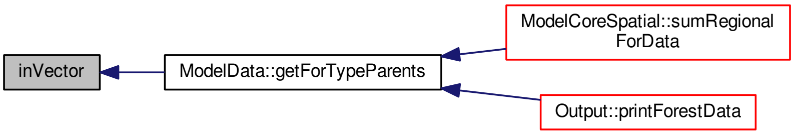
4.2.3.21 bool inVector (const K & *element*, const vector< K > & *v*) [inline]

Definition at line 426 of file BaseClass.h.

Referenced by ModelData::getForTypeParents().

```
00426                                                                               {
00427      for(unsigned int i=0; i<v.size(); i++){
00428        if(v[i]== element) return true;
00429      }
00430      return false;
00431   }
```

Here is the caller graph for this function:


inVector
ModelData::getForTypeParents
ModelCoreSpatial::sumRegional ForData
Output::printForestData

4.2.3.22 void msgOut (const int & *msgCode_h*, const string & *msg_h*, const bool & *refreshGUI_h* = true) const

Overloaded function to print the output log.

Overloaded method for the output log:

Parameters

msgCode_h	MSG_DEBUG, MSG_INFO, MSG_WARNING, MSG_ERROR, MSG_CRITICAL_ERROR
msg_h	message text (string)
refreshGUI_h	use this call to "ping" the GUI (optional, default=true)

Definition at line 50 of file BaseClass.cpp.

Referenced by Layers::addLegendItem(), Layers::addLegendItems(), Carbon::addSavedEmissions(), ModelData::addSetting(), ModelData::applyOverrides(), ModelCoreSpatial::assignSpMultiplierPropToVols(), ModelCoreSpatial::cachePixelExogenousData(), ModelCore::cacheSettings(), ModelCoreSpatial::cacheSettings(), Pixel::changeValue(), LLData::clean(), Output::cleanScenario(), ModelCore::computeCumulativeData(), ModelCoreSpatial::computeCumulativeData(), ModelCore::computeInventary(), ModelCoreSpatial::computeInventary(), Layers::countMyPixels(), ModelData::delDir(), Layers::filterExogenousDataset(), ModelRegion::getArea(), ModelData::getBaseData(), InputNode::getBoolContent(), Carbon::getCumSavedEmissions(), LLData::getData(), InputNode::getDoubleAttributeByName(), Pixel::getDoubleValue(), ModelData::getFilenamesByDir(), ModelData::getForData(), ModelData::getForType(), ModelData::getForTypeCounter(), ModelData::getForTypeParentId(), InputNode::getIntAttributeByName(), ModelData::getMaxYearUsableDeathTimber(), Pixel::getMyRegion(), InputNode::getNodeByName(), InputNode::getNodesByName(), Output::getOutputFieldDelimiter(), Pixel::getPastRegArea(), Pixel::getPixelsAtDistLevel(), ModelData::getProdData(), ModelData::getRegion(), ModelData::getScenarioIndex(), Pixel::getSpModifier(), Carbon::getStock(), InputNode::getStringAttributeByName(), ModelData::getTable(), ModelData::getTimedData(), ModelRegion::getValue(), ModelData::getVectorBaseData(), Output::initCarbonBalance(), Output::initDebugOutput(), Output::initDebugPixelValues(), Carbon::initialiseDeathBiomassStocks(), Carbon::initialiseProductsStocks(), ModelCoreSpatial::initializePixelArea(), ModelCoreSpatial::initializePixelVolumes(), ModelCore::initMarketModule(), ModelCoreSpatial::initMarketModule(), Output::initOptimisationLog(), Output::initOutputForestData(), Output::initOutputMaps(), Output::initOutputProductData(), ModelData::loadDataFromCache(), ModelData::loadInput(), Layers::print(), Output::printCarbonBalance(), Output::printDebugOutput(), Output::printDebugPixelValues(), Output::printForestData(), Output::printOptLog(), Output::printProductData(), ModelData::regSName2RegId(), ModelCoreSpatial::resetPixelValues(), Scheduler::run(), ModelCore::runBiologicalModule(), ModelCoreSpatial::runBiologicalModule(), ModelCore::runManagementModule(), ModelCoreSpatial::runManagementModule(), ModelCore::runMarketModule(), ModelCoreSpatial::runMarketModule(), ModelData::setBasicData(), ModelData::setDefaultForData(), ModelData::setDefaultPathogenRules(), ModelData::setDefaultProdData(), ModelData::setDefaultProductResourceMatrixLink(), ModelData::setDefaultSettings(), ModelData::setForData(), Init::setInitLevel(), Init::setInitLevel1(), Init::setInitLevel6(), Pixel::setPastRegArea(), ModelData::setProdData(), ModelData::setReclassificationRules(), ModelData::setTimedData(), InputNode::setWorkingFile(), ModelCoreSpatial::sumRegionalForData(), Pixel::swap(), ModelData::unpackKeyForData(), ModelData::unpackKeyProdData(), ModelCore::updateMapAreas(), and ModelCoreSpatial::updateMapAreas().

```
00050                                                                                                     {
00051
00052   msgOut2(msgCode_h, msg_h, refreshGUI_h);
00053
00054 }
```

4.2.3.23 void msgOut (const int & *msgCode_h,* const int & *msg_h,* const bool & *refreshGUI_h* = `true`) const

Overloaded function to print the output log.

Overloaded method for the output log:

Parameters

msgCode_h	MSG_DEBUG, MSG_INFO, MSG_WARNING, MSG_ERROR, MSG_CRITICAL_ERROR
msg_h	message text (int)
refreshGUI_h	use this call to "ping" the GUI (optional, default=true)

Definition at line 65 of file BaseClass.cpp.

```
00065                                                                                         {
00066   msgOut2(msgCode_h, i2s(msg_h), refreshGUI_h);
00067 }
```

4.2.3.24 void msgOut (const int & *msgCode_h*, const double & *msg_h*, const bool & *refreshGUI_h* = `true`) const

Overloaded function to print the output log.

Overloaded method for the output log:

Parameters

msgCode_h	MSG_DEBUG, MSG_INFO, MSG_WARNING, MSG_ERROR, MSG_CRITICAL_ERROR
msg_h	message text (double)
refreshGUI↩ _h	use this call to "ping" the GUI (optional, default=true)

Definition at line 78 of file BaseClass.cpp.

```
00078                                                                                            {
00079   msgOut2(msgCode_h, d2s(msg_h), refreshGUI_h);
00080
00081 }
```

4.2.3.25 void msgOut2 (const int & *msgCode_h*, const string & *msg_h*, const bool & *refreshGUI_h*) const [private]

Do the job of the overloaded functions.

Convenient (private) function to actually do the job of the overloaded functions

Definition at line 88 of file BaseClass.cpp.

```
00088                                                                                              {
00089
00090   string prefix;
00091   switch (msgCode_h){
00092   case MSG_NO_MSG:
00093     return;
00094   case MSG_DEBUG:
00095     prefix="*DEBUG: ";
00096     break;
00097   case MSG_INFO:
00098     prefix="**INFO: ";
00099     break;
00100   case MSG_WARNING:
00101     prefix="**WARNING: ";
00102     break;
00103   case MSG_ERROR:
00104     prefix="***ERROR: ";
00105     break;
00106   case MSG_CRITICAL_ERROR:
00107     prefix="****CRITICAL ERROR: ";
00108     break;
00109   default:
00110     cerr<<"I got an unknow error code: "<<msgCode_h<<" ("<<msg_h<<")"<<endl;
00111     exit(EXIT_FAILURE);
00112   }
00113
00114   string message = prefix+msg_h;
00115   if (MTHREAD && MTHREAD->usingGUI()){
00116     MTHREAD->msgOut(msgCode_h, message);
00117   }
00118   else {
00119     string totalMsg = prefix+msg_h;
00120     cout<< totalMsg <<endl;
00121   }
00122
00123   if(refreshGUI_h) {refreshGUI();}
00124
00125   if(msgCode_h==MSG_CRITICAL_ERROR){
00126     if (MTHREAD && MTHREAD->usingGUI()){
00127       throw(2);
00128     }
00129     else {
00130       //throw(2);
00131       exit(EXIT_FAILURE);
00132     }
00133   }
00134 }
```

4.2.3.26 double normSample (const double & *avg*, const double & *stdev*, const double & *minval* = `NULL`, const double & *maxval* = `NULL`) const

Sample from a normal distribution with bounds. Slower (double time, but still you see the diff only after milion of loops).

It doesn't require the normal_distribution to be passed to it, but due to including MTHREAD its definition can't be placed in the header and hence it can not be templated, so it works only with doubles.

Definition at line 325 of file BaseClass.cpp.

```
00325
         {
00326   if(minval != NULL  && maxval != NULL){
00327     if (maxval <= minval){
00328       msgOut(MSG_CRITICAL_ERROR,"Error in normSample: the maxvalue is lower than
       the minvalue");
00329     }
00330   }
00331   for(;;){
00332     normal_distribution<double> d(avg,stdev);
00333     double c = d(*MTHREAD->gen);
00334     if( (minval == NULL || c >= minval) && (maxval == NULL || c <= maxval) ){
00335       return c;
00336     }
00337   }
00338   return minval;
00339 }
```

4.2.3.27 K normSample (normal_distribution< K > & *d*, std::mt19937 & *gen*, const K & *minval* = `NULL`, const K & *maxval* = `NULL`) const `[inline]`

Sample from a normal distribution with bounds. Faster (half time) as the normal_distribution is made only once.

Definition at line 440 of file BaseClass.h.

```
00440
                                {
00441     if(minval != NULL  && maxval != NULL){
00442       if (maxval <= minval){
00443         msgOut(MSG_CRITICAL_ERROR,"Error in normSample: the maxvalue is lower than
       the minvalue");
00444       }
00445     }
00446     for(;;){
00447       K c = d(gen);
00448       if( (minval == NULL || c >= minval) && (maxval == NULL || c <= maxval) ){
00449         return c;
00450       }
00451     }
00452     return minval;
00453   }
```

4.2.3.28 vector< T > positionsToContent (const vector< T > & *vector_h*, const vector< int > & *positions*) [inline]

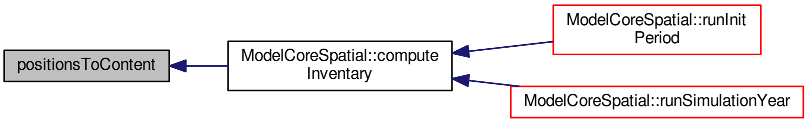
Return a vector of content from a vector and a vector of positions (int)

Definition at line 356 of file BaseClass.h.

Referenced by ModelCoreSpatial::computeInventary().

```
00356
      {
00357       vector <T> toReturn;
00358       for(uint i=0; i<positions.size(); i++){
00359           toReturn.push_back(vector_h.at(positions[i]));
00360       }
00361       return toReturn;
00362   }
```

Here is the caller graph for this function:


positionsToContent
ModelCoreSpatial::compute Inventary
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

4.2.3.29 void refreshGUI ( ) const

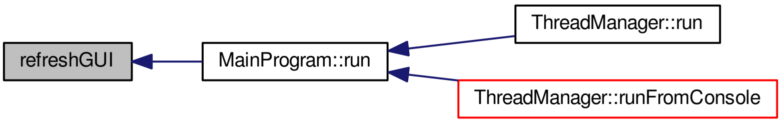
Ping to periodically return the control to the GUI.

Definition at line 137 of file BaseClass.cpp.

Referenced by MainProgram::run().

```
00137                            {
00138   if (MTHREAD && MTHREAD->usingGUI()){
00139     MTHREAD->refreshGUI();
00140   }
00141 }
```

Here is the caller graph for this function:



4.2.3.30 void resetMapValues (map< K, V > & *mymap*, const V & *value*) [inline]

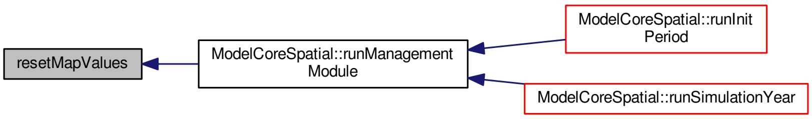
Reset all values stored in a map to the specified one.

Definition at line 338 of file BaseClass.h.

Referenced by ModelCoreSpatial::runManagementModule().

```
00338                                                                                                  {
00339       typename map<K, V>::iterator p;
00340       for(p=mymap.begin(); p!=mymap.end(); p++) {
00341           p->second =value;
00342       }
00343   }
```

Here is the caller graph for this function:


resetMapValues
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

4.2.3.31 bool s2b (const string & *string_h*) const

string to bool conversion

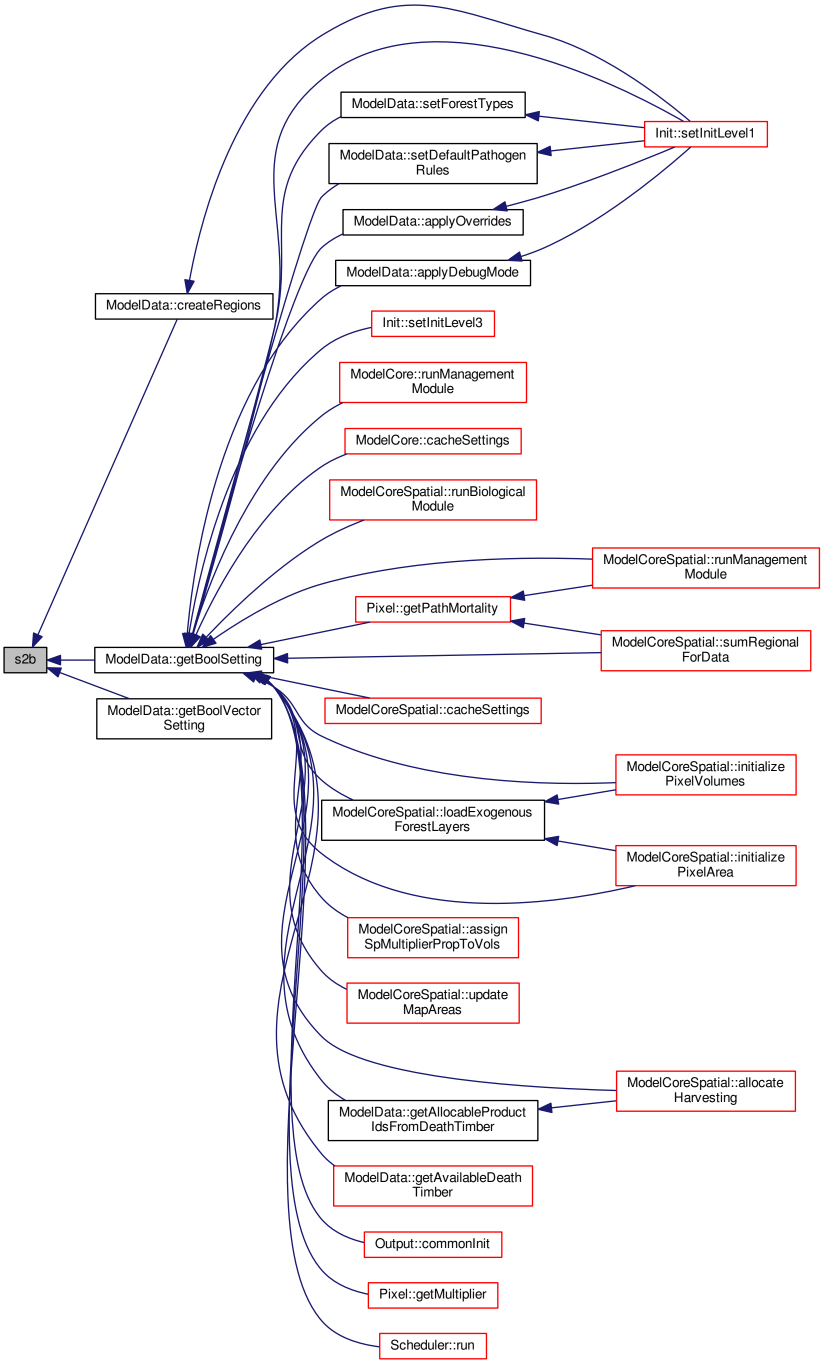
Includes conversion checks.

Definition at line 203 of file BaseClass.cpp.

Referenced by ModelData::createRegions(), ModelData::getBoolSetting(), and ModelData::getBoolVectorSetting().

```
00203                                             {
00204   if (string_h == "true" || string_h == "vero" || string_h == "TRUE" || string_h == "1" || string_h == "
      True")
00205     return true;
00206   else if (string_h == "false" || string_h == "falso" || string_h == "FALSE" || string_h == "0" || string_h
       == "" || string_h == "False")
00207     return false;
00208
00209   msgOut(MSG_CRITICAL_ERROR,"Conversion string to bool failed. Some problems with
       the data? (got\""+string_h+"\")");
00210   return true;
00211 }
```

Here is the caller graph for this function:


s2b
ModelData::createRegions
ModelData::getBoolSetting
ModelData::getBoolVectorSetting
ModelData::setForestTypes
ModelData::setDefaultPathogenRules
ModelData::applyOverrides
ModelData::applyDebugMode
Init::setInitLevel1
Init::setInitLevel3
ModelCore::runManagementModule
ModelCore::cacheSettings
ModelCoreSpatial::runBiologicalModule
ModelCoreSpatial::runManagementModule
Pixel::getPathMortality
ModelCoreSpatial::sumRegionalForData
ModelCoreSpatial::cacheSettings
ModelCoreSpatial::initializePixelVolumes
ModelCoreSpatial::loadExogenousForestLayers
ModelCoreSpatial::initializePixelArea
ModelCoreSpatial::assignSpMultiplierPropToVols
ModelCoreSpatial::updateMapAreas
ModelCoreSpatial::allocateHarvesting
ModelData::getAllocableProductIdsFromDeathTimber
ModelData::getAvailableDeathTimber
Output::commonInit
Pixel::getMultiplier
Scheduler::run

4.2.3.32 vector< bool > s2b (const vector< string > & *string_h*) const

string to bool conversion (vector)

Includes conversion checks.

Definition at line 267 of file BaseClass.cpp.

```
00267                                                   {
00268   vector <bool> valuesAsBool;
00269   for (uint i=0;i<string_h.size();i++){
00270     valuesAsBool.push_back(s2b(string_h[i]));
00271   }
00272   return valuesAsBool;
00273 }
```

4.2.3.33 double s2d (const string & *string_h*) const

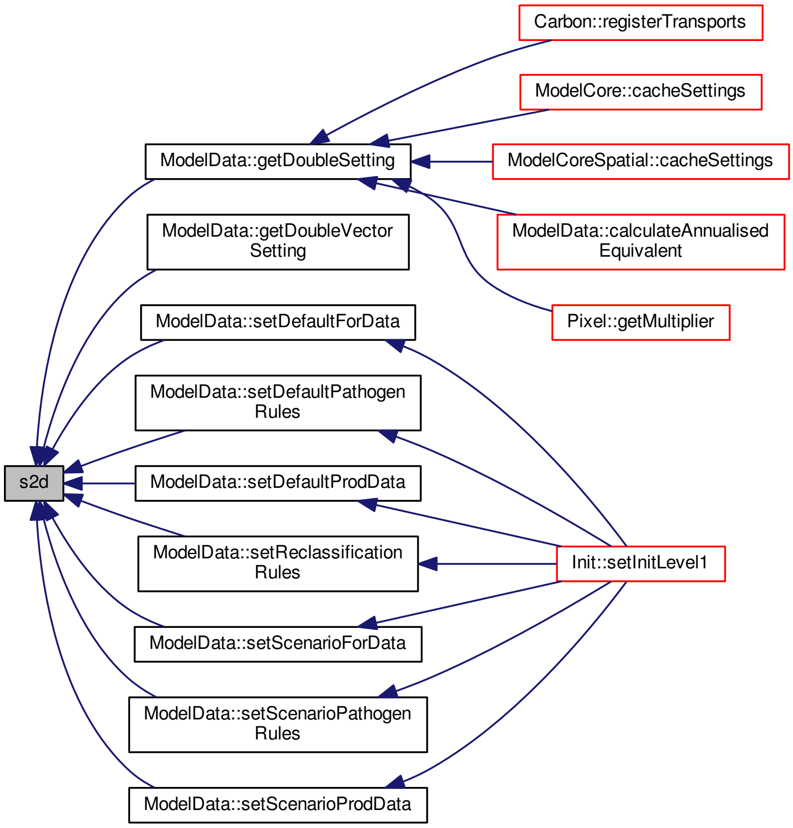
string to double conversion

Definition at line 166 of file BaseClass.cpp.

Referenced by ModelData::getDoubleSetting(), ModelData::getDoubleVectorSetting(), ModelData::setDefaultForData(), ModelData::setDefaultPathogenRules(), ModelData::setDefaultProdData(), ModelData::setReclassificationRules(), ModelData::setScenarioForData(), ModelData::setScenarioPathogenRules(), and ModelData::setScenarioProdData().

```
00166                                            {
00167   if (string_h == "") return 0.;
00168   double valueAsDouble;
00169   istringstream totalSString( string_h );
00170   totalSString >> valueAsDouble;
00171   return valueAsDouble;
00172   /*
00173   if (string_h == "") return 0.;
00174   try {
00175     return stod(string_h); // stod want dot as decimal separator in console mode and comma in gui mode.
       Further the decimal digits left are only 2 !!
00176   } catch (...) {
00177     if (string_h == "") return 0.;
00178     else {
00179       msgOut(MSG_CRITICAL_ERROR,"Conversion string to double failed. Some problems with the data?
       (got\""+string_h+"\")");
00180     }
00181   }
00182   return 0.;
00183   */
00184 }
```

Here is the caller graph for this function:


Carbon::registerTransports
ModelCore::cacheSettings
ModelData::getDoubleSetting
ModelCoreSpatial::cacheSettings
ModelData::getDoubleVector Setting
ModelData::calculateAnnualised Equivalent
ModelData::setDefaultForData
Pixel::getMultiplier
ModelData::setDefaultPathogen Rules
s2d
ModelData::setDefaultProdData
ModelData::setReclassification Rules
Init::setInitLevel1
ModelData::setScenarioForData
ModelData::setScenarioPathogen Rules
ModelData::setScenarioProdData

4.2.3.34 double s2d (const string & *string_h,* const bool & *replaceComma*) const

string to double conversion

Includes comma to dot conversion if needed.

Definition at line 189 of file BaseClass.cpp.

```
00189                                                                          {
00190   if(replaceComma){
00191     string valueAsString = string_h;
00192     // replace commas with dots. This is not needed when directly reading the input nodes as double, as the
       Qt function to Double does the same.
00193     replace(valueAsString.begin(), valueAsString.end(), ',', '.');
00194     return s2d(valueAsString);
00195   }
00196   return s2d(string_h);
00197   msgOut(MSG_CRITICAL_ERROR, "debug me please!");
00198   return 0.;
00199 }
```

4.2.3.35 vector< double > s2d (const vector< string > & *string_h*, const bool & *replaceComma* = `false`) const

string to double conversion (vector)

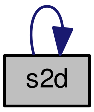
Includes comma to dot conversion if needed.

Definition at line 250 of file BaseClass.cpp.

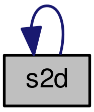
Referenced by s2d().

```
00250                                                                                {
00251   vector <double> valuesAsDouble;
00252   for (uint i=0;i<string_h.size();i++){
00253     if(replaceComma){
00254       string valueAsString = string_h[i];
00255       // replace commas with dots. This is not needed when directly reading the input nodes as double, as
       the Qt function to Double does the same.
00256       replace(valueAsString.begin(), valueAsString.end(), ',', '.');
00257       valuesAsDouble.push_back(s2d(valueAsString));
00258     } else {
00259       valuesAsDouble.push_back(s2d(string_h[i]));
00260     }
00261   }
00262   return valuesAsDouble;
00263 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.2.3.36 int s2i (const string & *string_h*) const

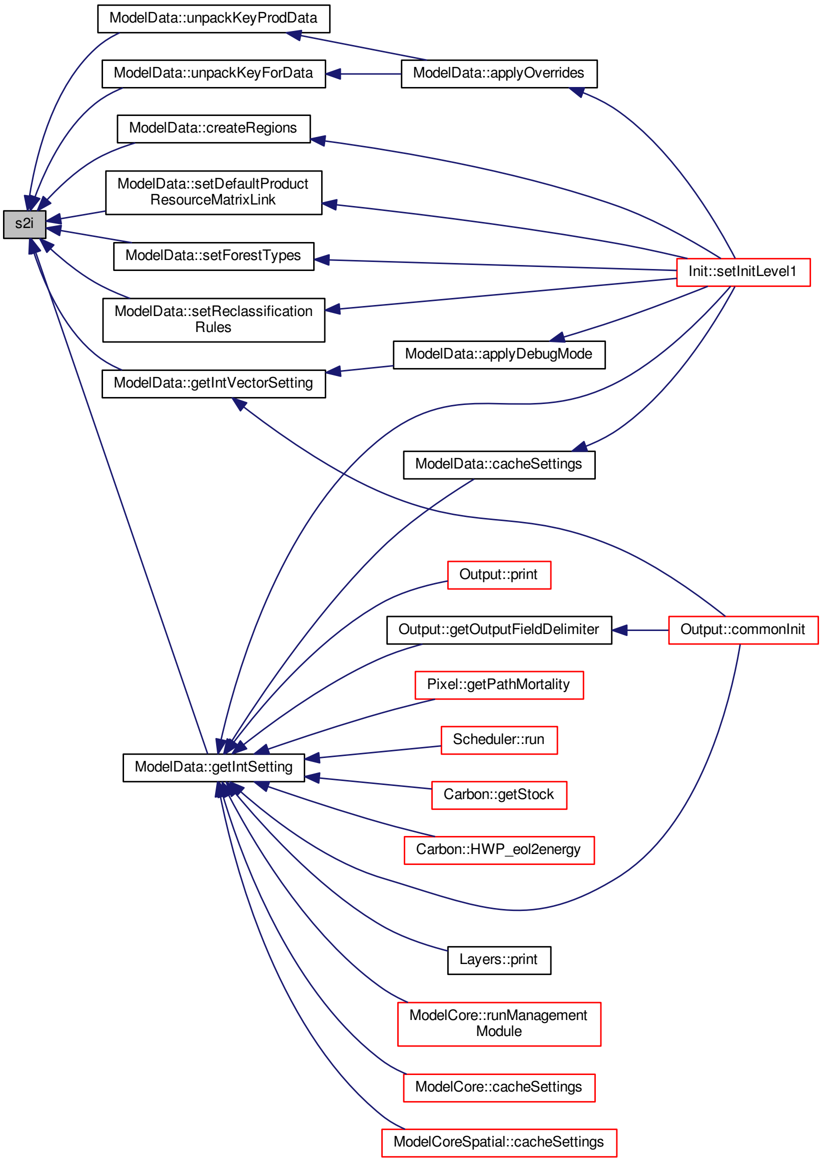
string to integer conversion

Definition at line 144 of file BaseClass.cpp.

Referenced by ModelData::createRegions(), ModelData::getIntSetting(), ModelData::getIntVectorSetting(), ModelData::setDefaultProductResourceMatrixLink(), ModelData::setForestTypes(), ModelData::setReclassificationRules(), ModelData::unpackKeyForData(), and ModelData::unpackKeyProdData().

```
00144                                            {
00145   if (string_h == "") return 0;
00146   int valueAsInteger;
00147   stringstream ss(string_h);
00148   ss >> valueAsInteger;
00149   return valueAsInteger;
00150   /*
00151   // I can’t use stoi as of bug in MinGW
00152   try {
00153     return stoi(string_h);
00154   } catch (...) {
00155     if (string_h == "") return 0;
00156     else {
00157       msgOut(MSG_CRITICAL_ERROR,"Conversion string to integer failed. Some problems with the data?
       (got\""+string_h+"\")");
00158     }
00159   }
00160   return 0;
00161   */
00162
00163 }
```

Here is the caller graph for this function:


ModelData::unpackKeyProdData
ModelData::unpackKeyForData
ModelData::applyOverrides
ModelData::createRegions
ModelData::setDefaultProduct ResourceMatrixLink
s2i
ModelData::setForestTypes
Init::setInitLevel1
ModelData::setReclassification Rules
ModelData::applyDebugMode
ModelData::getIntVectorSetting
ModelData::cacheSettings
Output::print
Output::getOutputFieldDelimiter
Output::commonInit
Pixel::getPathMortality
Scheduler::run
ModelData::getIntSetting
Carbon::getStock
Carbon::HWP_eol2energy
Layers::print
ModelCore::runManagement Module
ModelCore::cacheSettings
ModelCoreSpatial::cacheSettings

4.2.3.37 vector< int > s2i (const vector< string > & *string_h*) const

string to integer conversion (vector)

Definition at line 240 of file BaseClass.cpp.

```
                                                            {
        vector <int> valuesAsInteger;
        for (uint i=0;i<string_h.size();i++){
          valuesAsInteger.push_back(s2i(string_h[i]));
        }
        return valuesAsInteger;
      }
```

4.2.3.38 T stringTo (const std::string & *s*) const

Definition at line 343 of file BaseClass.cpp.

```
00343                                             {
00344   std::istringstream iss(s);
00345   T x;
00346   iss >> x;
00347   return x;
00348 }
```

4.2.3.39 void tokenize (const string & *str,* vector< string > & *tokens,* const string & *delimiter =* " ") const

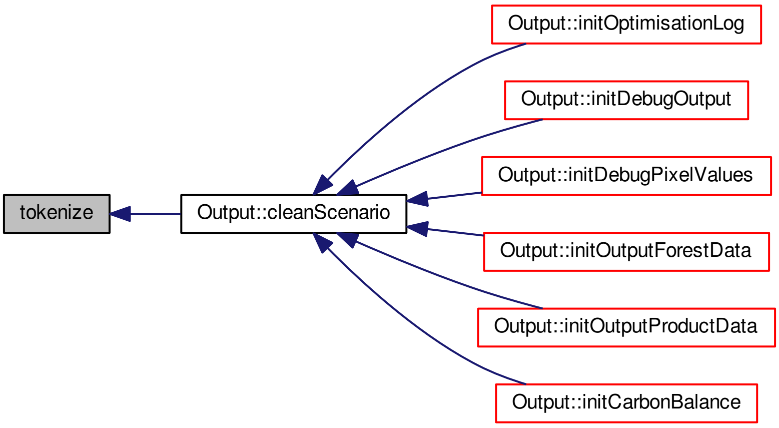
Tokenize a string using a delimiter (default is space)

Definition at line 369 of file BaseClass.cpp.

Referenced by Output::cleanScenario().

```
00369                                                                                                        {
00370     // Skip delimiters at beginning.
00371     string::size_type lastPos = str.find_first_not_of(delimiter, 0);
00372     // Find first "non-delimiter".
00373     string::size_type pos     = str.find_first_of(delimiter, lastPos);
00374 
00375     while (string::npos != pos || string::npos != lastPos)
00376     {
00377         // Found a token, add it to the vector.
00378         tokens.push_back(str.substr(lastPos, pos - lastPos));
00379         // Skip delimiters.  Note the "not_of"
00380         lastPos = str.find_first_not_of(delimiter, pos);
00381         // Find next "non-delimiter"
00382         pos = str.find_first_of(delimiter, lastPos);
00383     }
00384 }
```

Here is the caller graph for this function:


tokenize
Output::cleanScenario
Output::initOptimisationLog
Output::initDebugOutput
Output::initDebugPixelValues
Output::initOutputForestData
Output::initOutputProductData
Output::initCarbonBalance

4.2.3.40 string toString (const T & *x*) const

4.2.3.41 std::string toString (const T & *x*) const

Definition at line 317 of file BaseClass.cpp.

```
00317                                        {
00318   std::ostringstream oss;
00319   oss << x;
00320   return oss.str();
00321 }
```

4.2.3.42 void untokenize (string & *str,* vector< string > & *tokens,* const string & *delimiter* = " ") const

Definition at line 387 of file BaseClass.cpp.

```
00387                                                                                            {
00388   // add initial loken in str is not empty
00389   if(str != ""){
00390     str += delimiter;
00391   }
00392   for(int i=0;i<tokens.size();i++){
00393     str += tokens[i];
00394     // don't add final delimiter
00395     if(i != (tokens.size()-1)){
00396        str += delimiter;
00397     }
00398   }
00399 }
```

4.2.3.43 map< K, V > vectorToMap (const vector< K > & *keys,* const V & *value* = `0.0`) `[inline]`

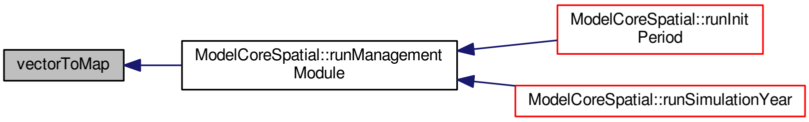
Returns a map built using the given vector and the given (scalar) value as keys/values pairs.

Definition at line 346 of file BaseClass.h.

Referenced by ModelCoreSpatial::runManagementModule().

```
00346                                                                                                          {
00347       map<K,V> returnMap;
00348       for(unsigned int i=0; i<keys.size();i++){
00349           pair<K,V> apair(keys[i],value);
00350           returnMap.insert(apair);
00351       }
00352       return returnMap;
00353   }
```

Here is the caller graph for this function:


vectorToMap
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

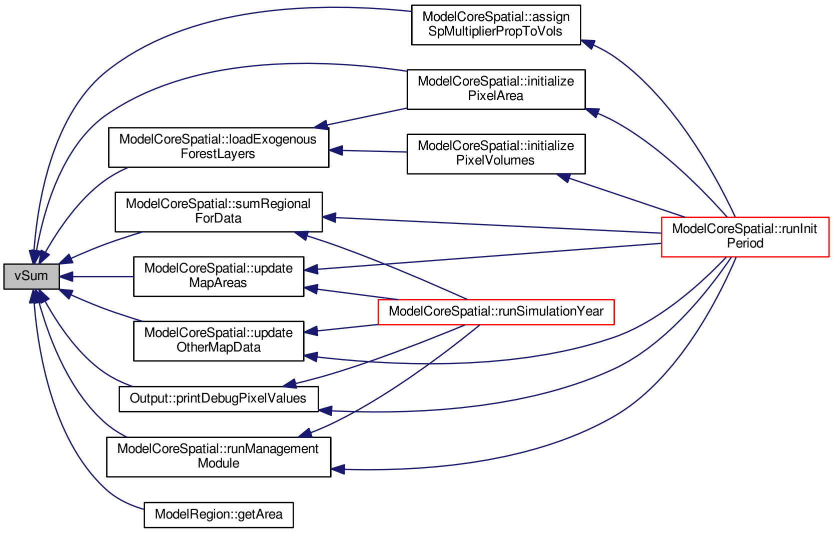
4.2.3.44 int vSum (const vector< int > & *vector_h*) const [inline]

Definition at line 273 of file BaseClass.h.

Referenced by ModelCoreSpatial::assignSpMultiplierPropToVols(), ModelRegion::getArea(), ModelCoreSpatial::initializePixelArea(), ModelCoreSpatial::loadExogenousForestLayers(), Output::printDebugPixelValues(), ModelCoreSpatial::runManagementModule(), ModelCoreSpatial::sumRegionalForData(), ModelCoreSpatial::updateMapAreas(), and ModelCoreSpatial::updateOtherMapData().

```
00273 {return accumulate(vector_h.begin(),vector_h.end(),0);};
```

Here is the caller graph for this function:


ModelCoreSpatial::assign SpMultiplierPropToVols
ModelCoreSpatial::initialize PixelArea
ModelCoreSpatial::loadExogenous ForestLayers
ModelCoreSpatial::initialize PixelVolumes
ModelCoreSpatial::sumRegional ForData
ModelCoreSpatial::runInit Period
vSum
ModelCoreSpatial::update MapAreas
ModelCoreSpatial::runSimulationYear
ModelCoreSpatial::update OtherMapData
Output::printDebugPixelValues
ModelCoreSpatial::runManagement Module
ModelRegion::getArea

4.2.3.45 double vSum (const vector< double > & *vector_h*) const [inline]

Definition at line 274 of file BaseClass.h.

```
00274 {return accumulate(vector_h.begin(),vector_h.end(),0.);};
```

4.2.3.46 int vSum (const vector< vector< int > > & *vector_h*) const

Definition at line 351 of file BaseClass.cpp.

```
00351                                                             {
00352   int toReturn = 0;
00353   for(vector < vector<int> >::const_iterator j=vector_h.begin();j!=vector_h.end();++j){
00354     toReturn += accumulate(j->begin(),j->end(),0);
00355   }
00356   return toReturn;
00357 }
```

4.2.3.47 double vSum (const vector< vector< double > > & *vector_h*) const

Definition at line 360 of file BaseClass.cpp.

```
00360                                                               {
00361   double toReturn = 0.0;
00362   for(vector < vector<double> >::const_iterator j=vector_h.begin();j!=vector_h.end();++j){
00363     toReturn += accumulate(j->begin(),j->end(),0.0);
00364   }
00365   return toReturn;
00366 }
```

4.2.4 Member Data Documentation

4.2.4.1 ThreadManager∗ MTHREAD [protected]

Pointer to the Thread manager.

Through this pointer each derived subclass (the vast maiority of those used on FFSM) can "ask" for sending signals to the GUI, like append the log or modify the map.

Definition at line 464 of file BaseClass.h.

Referenced by ModelCoreSpatial::allocateHarvesting(), ModelCoreSpatial::assignSpMultiplierPropToVols(), ModelCoreSpatial::cachePixelExogenousData(), ModelCore::cacheSettings(), ModelCoreSpatial::cacheSettings(), Carbon::Carbon(), Pixel::changeValue(), Output::commonInit(), ModelCore::computeCumulativeData(), ModelCoreSpatial::computeCumulativeData(), ModelCore::computeInventary(), ModelCoreSpatial::computeInventary(), Layers::countMyPixels(), ModelData::createRegions(), ModelData::getAllocableProductIdsFromDeathTimber(), ModelRegion::getArea(), ModelData::getAvailableAliveTimber(), ModelData::getBaseData(), ModelData::getBoolSetting(), ModelData::getBoolVectorSetting(), Layers::getCategory(), Layers::getColor(), ModelData::getDoubleSetting(), Pixel::getDoubleValue(), ModelData::getDoubleVectorSetting(), ModelData::getIntSetting(), ModelData::getIntVectorSetting(), Pixel::getMultiplier(), Output::getOutputFieldDelimiter(), Pixel::getPathMortality(), Pixel::getPixelsAtDistLevel(), ModelData::getRegionIds(), ModelData::getScenarioIndex(), Pixel::getSpModifier(), Carbon::getStock(), ModelData::getStringSetting(), ModelData::getStringVectorSetting(), ModelData::getTable(), ModelData::getTimedData(), ModelCore::gfd(), ModelCoreSpatial::gfd(), ModelCore::gpd(), ModelCoreSpatial::gpd(), Carbon::HWP_eol2energy(), Init::Init(), ModelCoreSpatial::initialiseCarbonModule(), Carbon::initialiseDeathBiomassStocks(), ModelCoreSpatial::initialiseDeathTimber(), Carbon::initialiseEmissionCounters(), Carbon::initialiseProductsStocks(), ModelCoreSpatial::initializePixelArea(), ModelCoreSpatial::initializePixelVolumes(), Layers::Layers(), LLData::LLData(), ModelData::loadDataFromCache(), ModelCoreSpatial::loadExogenousForestLayers(), ModelData::loadInput(), MainProgram::MainProgram(), ModelCore::ModelCore(), ModelCoreSpatial::ModelCoreSpatial(), ModelData::ModelData(), ModelRegion::ModelRegion(), Output::Output(), Pixel::Pixel(), Output::print(), Layers::print(), Layers::printBinMap(), Output::printCarbonBalance(), Output::printDebugOutput(), Output::printDebugPixelValues(), Output::printFinalOutput(), Output::printForestData(), Output::printMaps(), Output::printOptLog(), Output::printProductData(), Layers::randomShuffle(), ModelData::regId2RegSName(), ModelCoreSpatial::registerCarbonEvents(), Carbon::registerDeathBiomass(), Carbon::registerHarvesting(), Carbon::registerProducts(), Carbon::registerTransports(), ModelData::regSName2RegId(), ModelCoreSpatial::resetPixelValues(), Scheduler::run(), MainProgram::run(), ModelCore::runBiologicalModule(), ModelCoreSpatial::runBiologicalModule(), ModelCore::runInitPeriod(), ModelCoreSpatial::runInitPeriod(), ModelCore::runManagementModule(), ModelCoreSpatial::runManagementModule(), ModelCore::runMarketModule(), ModelCoreSpatial::runMarketModule(), ModelCore::runSimulationYear(), ModelCoreSpatial::runSimulationYear(), Scheduler::Scheduler(), ModelData::setDefaultSettings(), Init::setInitLevel1(), Init::setInitLevel3(), Init::setInitLevel5(), Init::setInitLevel6(), ModelRegion::setMyPixels(), ModelData::setOutputDirectory(), ModelData::setScenarioData(), ModelData::setTimedData(), ModelCore::sfd(), ModelCoreSpatial::sfd(), ModelCore::spd(), ModelCoreSpatial::spd(), ModelCoreSpatial::sumRegionalForData(), ModelCore::updateMapAreas(), ModelCoreSpatial::updateMapAreas(), and ModelCoreSpatial::updateOtherMapData().

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

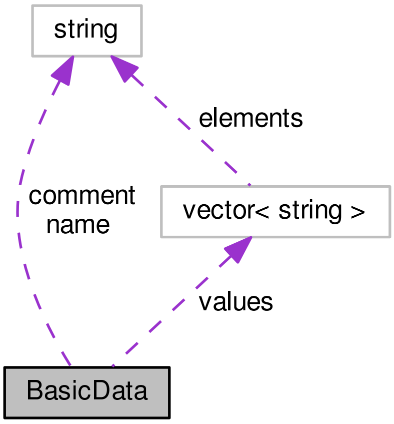
- /home/lobianco/git/ffsm_pp/src/BaseClass.h
- /home/lobianco/git/ffsm_pp/src/BaseClass.cpp

4.3 BasicData Struct Reference

Basic data units (struct)

`#include <ModelData.h>`

Collaboration diagram for BasicData:



Public Attributes

- string name
- vector< string > values

    Values are stored as "string" because we don't yet know at this point if they are string, double or integers!
- int type
- string comment

4.3.1 Detailed Description

Basic data units (struct)

Struct containing the basic data objects. At the moment, data are used to store programm settings or macro data.

Author

Antonello Lobianco

Definition at line 259 of file ModelData.h.

4.3.2 Member Data Documentation

4.3.2.1 string comment

Definition at line 264 of file ModelData.h.

Referenced by ModelData::addSetting(), and ModelData::setDefaultSettings().

4.3.2.2 string name

Definition at line 260 of file ModelData.h.

Referenced by ModelData::addSetting(), and ModelData::setDefaultSettings().

4.3.2.3 int type

Definition at line 263 of file ModelData.h.

Referenced by ModelData::addSetting(), and ModelData::setDefaultSettings().

4.3.2.4 vector< string > values

Values are stored as "string" because we don't yet know at this point if they are string, double or integers!

Definition at line 262 of file ModelData.h.

Referenced by ModelData::addSetting(), and ModelData::setDefaultSettings().

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

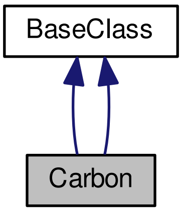
- /home/lobianco/git/ffsm_pp/src/ModelData.h

4.4 Carbon Class Reference

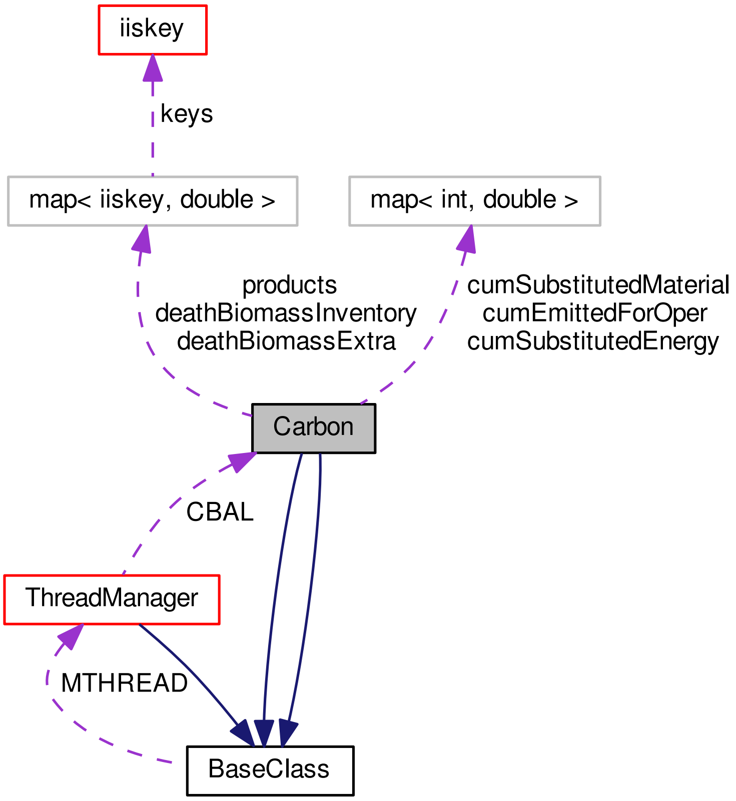
Class responsable to keep the logbook of the Carbon Balance.

`#include <Carbon.h>`

Inheritance diagram for Carbon:



Collaboration diagram for Carbon:



Public Member Functions

- Carbon (ThreadManager ∗MTHREAD_h)

  Constructor.
- ∼Carbon ()
- double getStock (const int &regId, const int &stock_type) const

  Returns the current stock of carbon [Mt CO2].
- double getCumSavedEmissions (const int &regId, const int &em_type) const

  Returns the current cumulative saved emissions by type [Mt CO2].
- void registerHarvesting (const double &value, const int &regId, const string &fType)

  Registers the harvesting of trees increasing the value of cumEmittedForOper.
- void registerDeathBiomass (const double &value, const int &regId, const string &fType)

  Registers the "death" of a given amount of biomass, storing it in the deathBiomass map.
- void registerProducts (const double &value, const int &regId, const string &productName)

  Registers the production of a given amount of products, storing it in the products maps. Also increase material substitution.
- void registerTransports (const double &distQ, const int &regId)

  Registers the quantities emitted by transport of wood FROM a given region.
- void initialiseDeathBiomassStocks (const vector< double > &deathByFt, const int &regId)

Initialises the stocks of death biomass for the avgLive_∗ years before the simulation starts.

- void initialiseProductsStocks (const vector< double > &qByProduct, const int &regId)

  Initialises the stocks of products for the avgLive_∗ years before the simulation starts.
- void initialiseEmissionCounters ()

  Initialises the emission counters to zero.
- void HWP_eol2energy ()

  Computes the energy substitution for the quota of HWP that reachs end of life and doesn't go to landfill.
- Carbon (ThreadManager ∗MTHREAD_h)

  Constructor.
- ∼Carbon ()
- double getStock (const int &regId, const int &stock_type) const

  Returns the current stock of carbon [Mt CO2].
- double getCumSavedEmissions (const int &regId, const int &em_type) const

  Returns the current cumulative saved emissions by type [Mt CO2].
- void registerHarvesting (const double &value, const int &regId, const string &fType)

  Registers the harvesting of trees increasing the value of cumEmittedForOper.
- void registerDeathBiomass (const double &value, const int &regId, const string &fType)

  Registers the "death" of a given amount of biomass, storing it in the deathBiomass map.
- void registerProducts (const double &value, const int &regId, const string &productName)

  Registers the production of a given amount of products, storing it in the products maps. Also increase material substitution.
- void registerTransports (const double &distQ, const int &regId)

  Registers the quantities emitted by transport of wood FROM a given region.
- void initialiseDeathBiomassStocks (const vector< double > &deathByFt, const int &regId)

  Initialises the stocks of death biomass for the avgLive_∗ years before the simulation starts.
- void initialiseProductsStocks (const vector< double > &qByProduct, const int &regId)

  Initialises the stocks of products for the avgLive_∗ years before the simulation starts.
- void initialiseEmissionCounters ()

  Initialises the emission counters to zero.
- void HWP_eol2energy ()

  Computes the energy substitution for the quota of HWP that reachs end of life and doesn't go to landfill.

Private Member Functions

- void addSavedEmissions (const double &value, const int &regId, const int &em_type)

  Increases the value to the saved emissions for a given type and region.
- double getRemainingStock (const double &initialValue, const double &halfLife, const double &years) const

  Apply a single exponential decay model to retrieve the remining stock given the initial stock, the half life and the time passed from stock formation.
- void addSavedEmissions (const double &value, const int &regId, const int &em_type)

  Increases the value to the saved emissions for a given type and region.
- double getRemainingStock (const double &initialValue, const double &halfLife, const double &years) const

  Apply a single exponential decay model to retrieve the remining stock given the initial stock, the half life and the time passed from stock formation.

Private Attributes

- map< iiskey, double > deathBiomassInventory

  Map that register the death of biomass by year, l2_region and forest type (inventoried)[Mm^3 wood].
- map< iiskey, double > deathBiomassExtra

  Map that register the death of biomass by year, l2_region and forest type (non-inventoried biomass: branches, roots..) [Mm^3 wood].
- map< iiskey, double > products

  Map that register the production of a given product by year, l2_region and product [Mm^3 wood].
- map< int, double > cumSubstitutedEnergy

  Map that store the cumulative CO2 substituted for energy consumption, by l2_region [Mt CO2].
- map< int, double > cumSubstitutedMaterial

  Map that store the cumulative CO2 substituted using less energivory material, by l2_region [Mt CO2].
- map< int, double > cumEmittedForOper

  Map that store emissions for forest operations, including transport, by l2_region [Mt CO2].

Additional Inherited Members

4.4.1 Detailed Description

Class responsable to keep the logbook of the Carbon Balance.

Class responsable to keep the logbook of the Death Timber still usable by the market module.

Author

Antonello Lobianco

A single istance of this class exists and is available trought the global MTHREAD->CBAL pointer.

It consits of functions to track a carbon-related event and store the information in STL maps that either register the events (for the stocks) or contain the cumulated carbon (for emission flows).

Carbon pools are stored as Mm^3 wood while and emission cumulated counters are directly in Mt CO2.

getStock() and getCumSavedEmissions() are then used to report the current levels of carbon in the stock or emitted/substituted.

Author

Antonello Lobianco

A single istance of this class exists and is available trought the global MTHREAD->MLB pointer.

It consits of functions to track a mortality-related event and store the information in STL maps that register the events and keep updated the stocks.

Carbon pools are stored as Mm^3 wood while and emission cumulated counters are directly in Mt CO2.

getStock() and getCumSavedEmissions() are then used to report the current levels of carbon in the stock or emitted/substituted.

Definition at line 50 of file Carbon.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Carbon (ThreadManager ∗ *MTHREAD_h*)

Constructor.

Definition at line 32 of file Carbon.cpp.

```
00032                                    {
00033   MTHREAD=MTHREAD_h;
00034 }
```

4.4.2.2 ∼Carbon ( )

Definition at line 36 of file Carbon.cpp.

```
00036                {
00037 }
```

4.4.2.3 Carbon (ThreadManager ∗ *MTHREAD_h*)

Constructor.

4.4.2.4 ∼Carbon ( )

4.4.3 Member Function Documentation

4.4.3.1 void addSavedEmissions (const double & *value,* const int & *regId,* const int & *em_type*) [private]

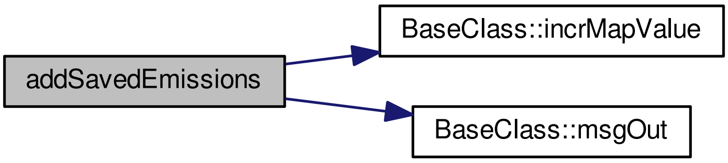
Increases the value to the saved emissions for a given type and region.

Definition at line 325 of file Carbon.cpp.

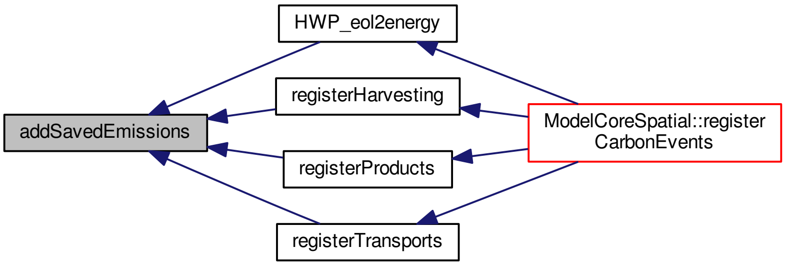
Referenced by HWP_eol2energy(), registerHarvesting(), registerProducts(), and registerTransports().

```
00325                                                                                    {
00326   switch (em_type){
00327     case EM_ENSUB:
00328       incrMapValue(cumSubstitutedEnergy, regId, value);
00329       break;
00330     case EM_MATSUB:
00331       incrMapValue(cumSubstitutedMaterial, regId, value);
00332       break;
00333     case EM_FOROP:
00334       incrMapValue(cumEmittedForOper, regId, -value);
00335       break;
00336     default:
00337       msgOut(MSG_CRITICAL_ERROR,"Unexpected em_type in function
      getCumSavedEmissions");
00338   }
00339 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.3.2 void addSavedEmissions (const double & *value,* const int & *regId,* const int & *em_type*) [private]

Increases the value to the saved emissions for a given type and region.

4.4.3.3 double getCumSavedEmissions (const int & *regId,* const int & *em_type*) const

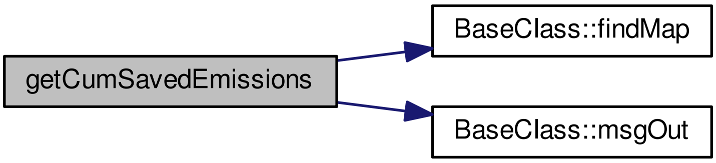
Returns the current cumulative saved emissions by type [Mt CO2].

Definition at line 138 of file Carbon.cpp.

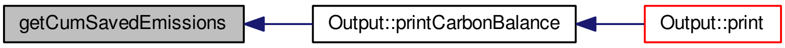
Referenced by Output::printCarbonBalance().

```
00138                                                                          {
00139   switch (em_type){
00140     case EM_ENSUB:
00141       return findMap(cumSubstitutedEnergy, regId);
00142       break;
00143     case EM_MATSUB:
00144       return findMap(cumSubstitutedMaterial, regId);
00145       break;
00146     case EM_FOROP:
00147       return -findMap(cumEmittedForOper, regId);
00148       break;
00149     default:
00150       msgOut(MSG_CRITICAL_ERROR,"Unexpected em_type in function
       getCumSavedEmissions");
00151   }
00152   return 0.0;
00153 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.3.4 double getCumSavedEmissions (const int & *regId,* const int & *em_type*) const

Returns the current cumulative saved emissions by type [Mt CO2].

4.4.3.5 double getRemainingStock (const double & *initialValue,* const double & *halfLife,* const double & *years*) const [private]

Apply a single exponential decay model to retrieve the remining stock given the initial stock, the half life and the time passed from stock formation.

4.4.3.6 double getRemainingStock (const double & *initialValue,* const double & *halfLife,* const double & *years*) const [private]

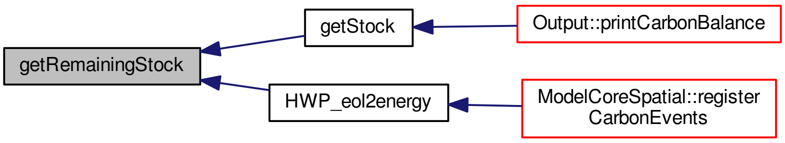
Apply a single exponential decay model to retrieve the remining stock given the initial stock, the half life and the time passed from stock formation.

Definition at line 342 of file Carbon.cpp.

Referenced by getStock(), and HWP_eol2energy().

```
00342                                                                                                          {
00343   // // TODO: remove this test
00344   //if(years>0) return 0.0;
00345   //return initialValue;
00346 
00347   double k  = log(2)/halfLife;
00348   return initialValue*exp(-k*years);
00349 }
```

Here is the caller graph for this function:



4.4.3.7 double getStock (const int & *regId,* const int & *stock_type*) const

Returns the current stock of carbon [Mt CO2].

Parameters

reg	
stock_type	

Returns

the Carbon stocked in a given sink

For product sink:

- for primary products it includes the primary products exported out of the country, but not those exported to other regions or used in the region as these are assumed to be totally transformed to secondary products;
- for secondary products it includes those produced in the region from locally or regionally imported primary product plus those secondary products imported from other regions, less those exported to other regions. It doesn't include the secondary products imported from abroad the country.

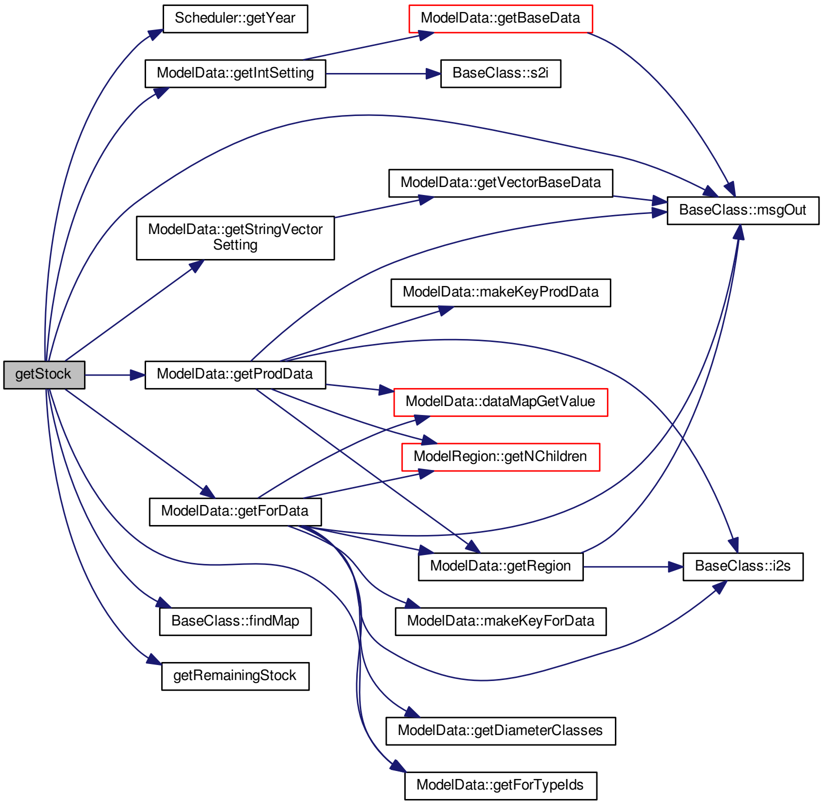
Definition at line 53 of file Carbon.cpp.

Referenced by Output::printCarbonBalance().

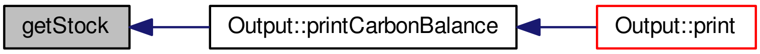
```
00053                                                                 {
00054   double toReturn = 0.0;
00055   int currentYear = MTHREAD->SCD->getYear();
00056   int initialYear = MTHREAD->MD->getIntSetting("initialYear");
00057   switch (stock_type){
00058     case STOCK_PRODUCTS: {
00059       vector <string> priProducts = MTHREAD->MD->getStringVectorSetting("
      priProducts");
00060       vector <string> secProducts = MTHREAD->MD->getStringVectorSetting("
      secProducts");
00061       vector <string> allProducts = priProducts;
00062       allProducts.insert( allProducts.end(), secProducts.begin(), secProducts.end() );
00063       for(uint i=0;i<allProducts.size();i++){
00064         double coeff = MTHREAD->MD->getProdData("co2content_products",regId,allProducts
      [i],DATA_NOW,""); //  [kg CO2/m^3 wood]
00065         double life  = MTHREAD->MD->getProdData("avgLife_products",regId,allProducts[i]
      ,DATA_NOW,"");
00066         //for(int y=currentYear;y>currentYear-life;y--){ // ok
00067         //  iiskey key(y,regId,allProducts[i]);
00068         //  toReturn += findMap(products,key,MSG_NO_MSG,0.0)*coeff/1000;
00069         //}
```

```
        for(int y=(initialYear-100);y<=currentYear;y++){
          iiskey key(y,regId,allProducts[i]);
          double originalStock = findMap(products,key,MSG_NO_MSG,0.0);
          double remainingStock = getRemainingStock(originalStock,life,currentYear-y);
          toReturn += remainingStock*coeff/1000;
        }
      }
      break;
    }
    case STOCK_INV:{
      vector <string> fTypes = MTHREAD->MD->getForTypeIds();
      for(uint i=0;i<fTypes.size();i++){
        // units:
        // co2content_inventory: [Kg CO2 / m^3 wood]
        // co2content_extra:     [Kg CO2 / m^3 inventaried wood]
        double coeff = MTHREAD->MD->getForData("co2content_inventory",regId,fTypes[i],"",DATA_NOW); //  [kg CO2/m^3 wood]
        double life  = MTHREAD->MD->getForData("avgLive_deathBiomass_inventory",regId,fTypes[i],"",DATA_NOW);
        // PART A: from death biomass..
        //for(int y=currentYear;y>currentYear-life;y--){ // ok
        //  iiskey key(y,regId,fTypes[i]);
        //  toReturn += findMap(deathBiomassInventory,key,MSG_NO_MSG)*coeff/1000;
        //}
        for(int y=(initialYear-100);y<=currentYear;y++){
          iiskey key(y,regId,fTypes[i]);
          double originalStock = findMap(deathBiomassInventory,key,MSG_NO_MSG,0.0);
          double remainingStock = getRemainingStock(originalStock,life,currentYear-y);
          toReturn += remainingStock*coeff/1000;
        }

        // PART B: from inventory volumes
        toReturn += MTHREAD->MD->getForData("vol",regId,fTypes[i],DIAM_ALL,DATA_NOW)*coeff/1000;
      }
      break;

    }
    case STOCK_EXTRA:{
      vector <string> fTypes = MTHREAD->MD->getForTypeIds();
      for(uint i=0;i<fTypes.size();i++){
        // units:
        // co2content_inventory: [Kg CO2 / m^3 wood]
        // co2content_extra:     [Kg CO2 / m^3 inventaried wood]
        double coeff = MTHREAD->MD->getForData("co2content_extra",regId,fTypes[i],"",DATA_NOW); //  [kg CO2/m^3 wood]
        double life  = MTHREAD->MD->getForData("avgLive_deathBiomass_extra",regId,fTypes[i],"",DATA_NOW);
        // PART A: from death biomass..
        //for(int y=currentYear;y>currentYear-life;y--){ // ok
        //  iiskey key(y,regId,fTypes[i]);
        //  toReturn += findMap(deathBiomassExtra,key,MSG_NO_MSG),0.0*coeff/1000;
        //}
        for(int y=(initialYear-100);y<=currentYear;y++){
          iiskey key(y,regId,fTypes[i]);
          double originalStock = findMap(deathBiomassExtra,key,MSG_NO_MSG,0.0);
          double remainingStock = getRemainingStock(originalStock,life,currentYear-y);
          toReturn += remainingStock*coeff/1000;
        }
        // PART B: from inventory volumes
        double extraBiomass_ratio = MTHREAD->MD->getForData("extraBiomass_ratio",regId,fTypes[i],"",DATA_NOW);
        toReturn += MTHREAD->MD->getForData("vol",regId,fTypes[i],DIAM_ALL,DATA_NOW)*extraBiomass_ratio*coeff/1000;
      }
      break;
    }
    default:
      msgOut(MSG_CRITICAL_ERROR,"Unexpected stock_type in function getStock");
  }
  return toReturn;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.4.3.8 double getStock (const int & *regId,* const int & *stock_type*) const

Returns the current stock of carbon [Mt CO2].

4.4.3.9 void HWP_eol2energy ( )

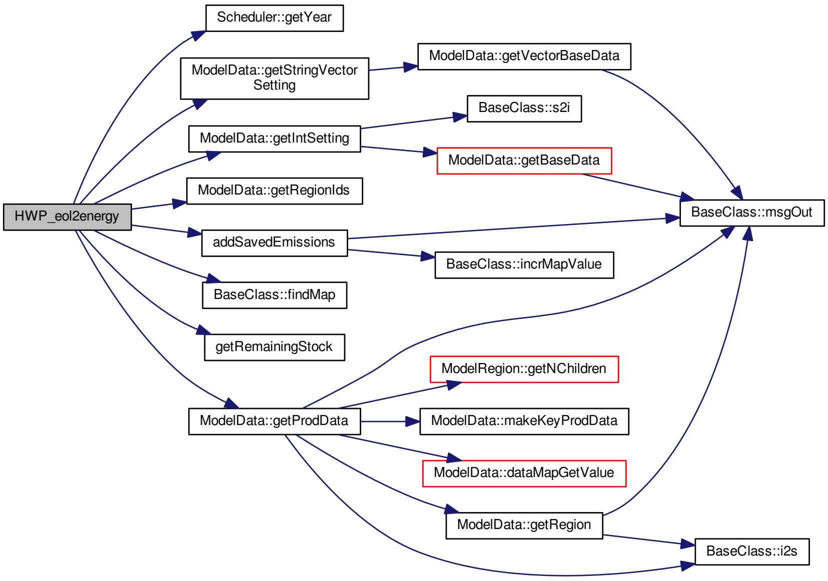
Computes the energy substitution for the quota of HWP that reachs end of life and doesn't go to landfill.

Definition at line 289 of file Carbon.cpp.

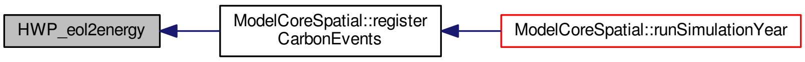
Referenced by ModelCoreSpatial::registerCarbonEvents().

```
                      {

  int currentYear = MTHREAD->SCD->getYear();
  int initialYear = MTHREAD->MD->getIntSetting("initialYear");
  vector <string> priProducts = MTHREAD->MD->getStringVectorSetting("priProducts");
  vector <string> secProducts = MTHREAD->MD->getStringVectorSetting("secProducts");
  vector <string> allProducts = priProducts;
  allProducts.insert( allProducts.end(), secProducts.begin(), secProducts.end() );

  vector<int> regIds = MTHREAD->MD->getRegionIds(2);
  for (uint r=0;r<regIds.size();r++){
    double regId = regIds[r];
    for(uint i=0;i<allProducts.size();i++){
      string pr = allProducts[i];
      double life  = MTHREAD->MD->getProdData("avgLife_products",regId,pr,DATA_NOW,"");
      double eol2e_share      = MTHREAD->MD->getProdData("eol2e_share",regId,pr,DATA_NOW,"");
      double subEnergyCoeff   = MTHREAD->MD->getProdData("co2sub_energy",regId,pr,DATA_NOW,"");
      if(eol2e_share > 0 && subEnergyCoeff>0){
        for(int y=(initialYear-100);y<currentYear;y++){ // notice the minor operator and not minor equal: energy substitution for products produced this year assigned to the following year, otherwise double counring in the process of making dicrete the exponential function
          iiskey key(y,regId,pr);
          double originalStock = findMap(products,key,MSG_NO_MSG,0.0);
          double remainingStockLastYear = getRemainingStock(originalStock,life,currentYear-y-1);
          double remainingStockThisYear = getRemainingStock(originalStock,life,currentYear-y);
          double eofThisYear = remainingStockLastYear-remainingStockThisYear;
          addSavedEmissions(subEnergyCoeff*eofThisYear*eol2e_share/1000,regId,EM_ENSUB);
        }
      }
    }
  }

}
```

Here is the call graph for this function:



Here is the caller graph for this function:


HWP_eol2energy
ModelCoreSpatial::register CarbonEvents
ModelCoreSpatial::runSimulationYear

4.4.3.10 void HWP_eol2energy ( )

Computes the energy substitution for the quota of HWP that reachs end of life and doesn't go to landfill.

4.4.3.11 void initialiseDeathBiomassStocks (const vector< double > & *deathByFt,* const int & *regId*)

Initialises the stocks of death biomass for the avgLive_∗ years before the simulation starts.

4.4.3.12 void initialiseDeathBiomassStocks (const vector< double > & *deathByFt,* const int & *regId*)

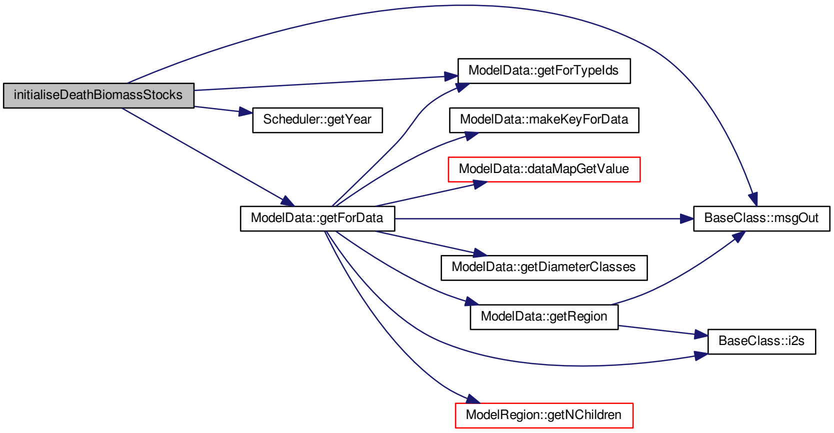
Initialises the stocks of death biomass for the avgLive_∗ years before the simulation starts.

Definition at line 169 of file Carbon.cpp.

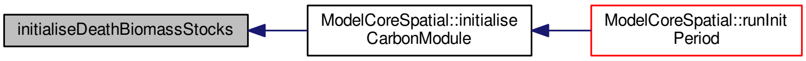
Referenced by ModelCoreSpatial::initialiseCarbonModule().

```
00169                                                                                       {
00170   // it must initialize in the past the death biomass taking the value of the first year
00171   vector <string> fTypes = MTHREAD->MD->getForTypeIds();
00172   if(fTypes.size() != deathByFt.size()) {msgOut(MSG_CRITICAL_ERROR,"deathByFt and
       fTypes have different lenght!");}
00173   int currentYear = MTHREAD->SCD->getYear();
00174   //int initialYear = MD->getIntSetting("initialYear");
00175
00176   for(uint i=0;i<fTypes.size();i++){
00177 //    double life_inventory     =
       MTHREAD->MD->getForData("avgLive_deathBiomass_inventory",regId,fTypes[i],"",DATA_NOW);
00178 //    double life_extra         =
       MTHREAD->MD->getForData("avgLive_deathBiomass_extra",regId,fTypes[i],"",DATA_NOW);
00179       double extraBiomass_ratio = MTHREAD->MD->getForData("extraBiomass_ratio",regId,
      fTypes[i],"",DATA_NOW);
00180
00181 //    for(int y=currentYear;y>currentYear-life_inventory;y--){
00182 //        iiskey key(y,regId,fTypes[i]);
00183 //        pair<iiskey,double> mypair(key,deathByFt.at(i));
00184 //        deathBiomassInventory.insert(mypair);
00185 //    }
00186 //    for(int y=currentYear;y>currentYear-life_extra;y--){
00187 //        iiskey key(y,regId,fTypes[i]);
00188 //        pair<iiskey,double> mypair(key,deathByFt.at(i)*extraBiomass_ratio);
00189 //        deathBiomassExtra.insert(mypair);
00190 //    }
00191
00192     for(int y=currentYear;y>currentYear-100;y--){
00193         iiskey key(y,regId,fTypes[i]);
00194         pair<iiskey,double> mypairInventory(key,deathByFt.at(i));
00195         pair<iiskey,double> mypairExtra(key,deathByFt.at(i)*extraBiomass_ratio);
00196         deathBiomassInventory.insert(mypairInventory);
00197         deathBiomassExtra.insert(mypairExtra);
00198     }
00199   }
00200 }
```

Here is the call graph for this function:


initialiseDeathBiomassStocks
ModelData::getForTypeIds
Scheduler::getYear
ModelData::makeKeyForData
ModelData::dataMapGetValue
ModelData::getForData
BaseClass::msgOut
ModelData::getDiameterClasses
ModelData::getRegion
BaseClass::i2s
ModelRegion::getNChildren

Here is the caller graph for this function:


initialiseDeathBiomassStocks
ModelCoreSpatial::initialise CarbonModule
ModelCoreSpatial::runInit Period

4.4.3.13 void initialiseEmissionCounters ( )

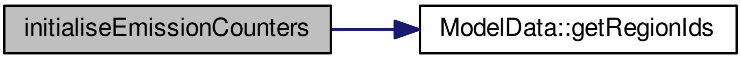
Initialises the emission counters to zero.

Definition at line 158 of file Carbon.cpp.

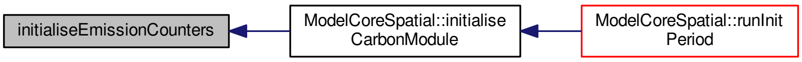
Referenced by ModelCoreSpatial::initialiseCarbonModule().

```
00158                                       {
00159   vector<int> regIds = MTHREAD->MD->getRegionIds(2);
00160   for (uint i=0;i<regIds.size();i++){
00161     pair<int,double> mypair(regIds[i],0.0);
00162     cumSubstitutedEnergy.insert(mypair);
00163     cumSubstitutedMaterial.insert(mypair);
00164     cumEmittedForOper.insert(mypair);
00165   }
00166 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


initialiseEmissionCounters
ModelCoreSpatial::initialise CarbonModule
ModelCoreSpatial::runInit Period

4.4.3.14 void initialiseEmissionCounters ( )

Initialises the emission counters to zero.

4.4.3.15 void initialiseProductsStocks (const vector< double > & *qByProduct,* const int & *regId*)

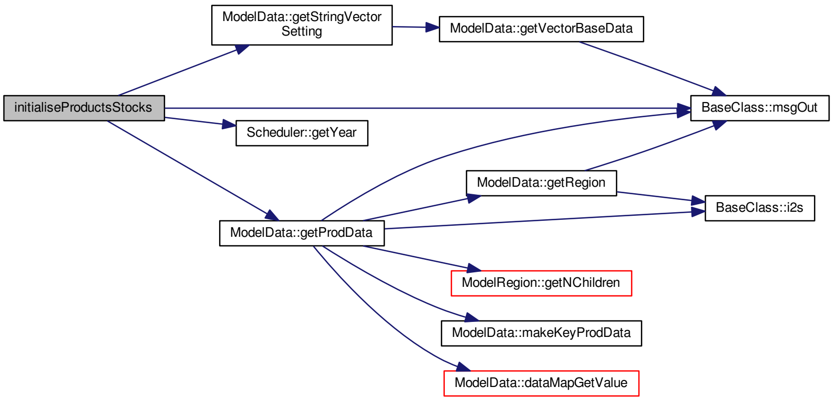
Initialises the stocks of products for the avgLive_∗ years before the simulation starts.

Definition at line 203 of file Carbon.cpp.

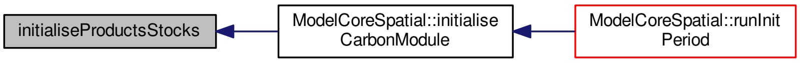
Referenced by ModelCoreSpatial::initialiseCarbonModule().

```
00203                                                                                {
00204   // it must initialize in the past the products taking the value of the first year
00205   vector <string> priProducts = MTHREAD->MD->getStringVectorSetting("
      priProducts");
00206   vector <string> secProducts = MTHREAD->MD->getStringVectorSetting("
      secProducts");
00207   vector <string> allProducts = priProducts;
00208   allProducts.insert( allProducts.end(), secProducts.begin(), secProducts.end() );
00209   if(allProducts.size() != qByProduct.size()) {msgOut(MSG_CRITICAL_ERROR,"
      allProducts and qByProduct have different lenght!");}
00210   int currentYear = MTHREAD->SCD->getYear();
00211   for(uint i=0;i<allProducts.size();i++){
00212     double life  = MTHREAD->MD->getProdData("avgLife_products",regId,allProducts[i],
      DATA_NOW);
00213     //for(int y=currentYear;y>currentYear-life;y--){
00214     for(int y=currentYear;y>currentYear-100;y--){
00215         iiskey key(y,regId,allProducts[i]);
00216         pair<iiskey,double> mypair(key,qByProduct.at(i));
00217         products.insert(mypair);
00218     }
00219   }
00220   //cout << "" << endl;
00221 }
```

Here is the call graph for this function:


ModelData::getStringVector Setting
ModelData::getVectorBaseData
initialiseProductsStocks
BaseClass::msgOut
Scheduler::getYear
ModelData::getRegion
BaseClass::i2s
ModelData::getProdData
ModelRegion::getNChildren
ModelData::makeKeyProdData
ModelData::dataMapGetValue

Here is the caller graph for this function:


initialiseProductsStocks
ModelCoreSpatial::initialise CarbonModule
ModelCoreSpatial::runInit Period

4.4.3.16 void initialiseProductsStocks (const vector< double > & *qByProduct,* const int & *regId*)

Initialises the stocks of products for the avgLive_∗ years before the simulation starts.

4.4.3.17 void registerDeathBiomass (const double & *value,* const int & *regId,* const string & *fType*)

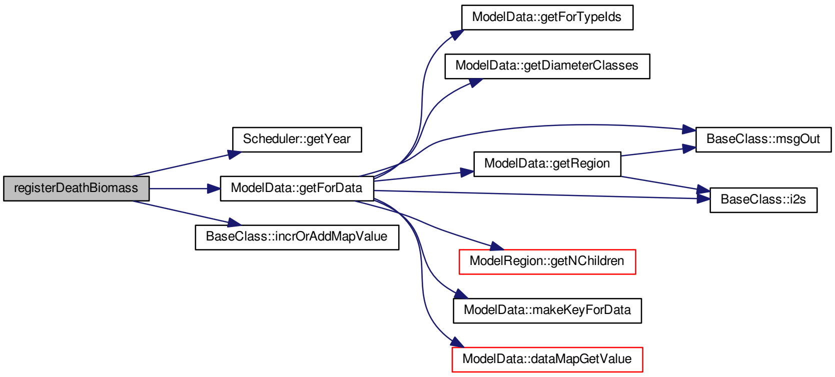
Registers the "death" of a given amount of biomass, storing it in the deathBiomass map.

Definition at line 243 of file Carbon.cpp.

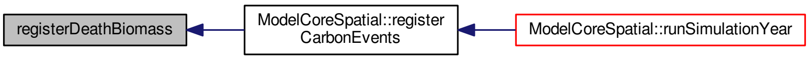
Referenced by ModelCoreSpatial::registerCarbonEvents().

```
00243                                                                                 {
00244   int year = MTHREAD->SCD->getYear();
00245   iiskey key(year,regId,fType);
00246   double extraBiomass_ratio = MTHREAD->MD->getForData("extraBiomass_ratio",regId,fType,"
      ",DATA_NOW);
00247   //pair<iiskey,double> mypairInventory(key,value);
00248   //pair<iiskey,double> mypairExtra(key,value*extraBiomass_ratio);
00249   incrOrAddMapValue(deathBiomassInventory, key, value);
00250   incrOrAddMapValue(deathBiomassExtra, key, value*extraBiomass_ratio);
00251   //deathBiomassInventory.insert(mypairInventory);
00252   //deathBiomassExtra.insert(mypairExtra);
00253 
00254 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


registerDeathBiomass
ModelCoreSpatial::register CarbonEvents
ModelCoreSpatial::runSimulationYear

4.4.3.18 void registerDeathBiomass (const double & *value,* const int & *regId,* const string & *fType*)

Registers the "death" of a given amount of biomass, storing it in the deathBiomass map.

4.4.3.19 void registerHarvesting (const double & *value,* const int & *regId,* const string & *fType*)

Registers the harvesting of trees increasing the value of cumEmittedForOper.

4.4.3.20 void registerHarvesting (const double & *value,* const int & *regId,* const string & *fType*)

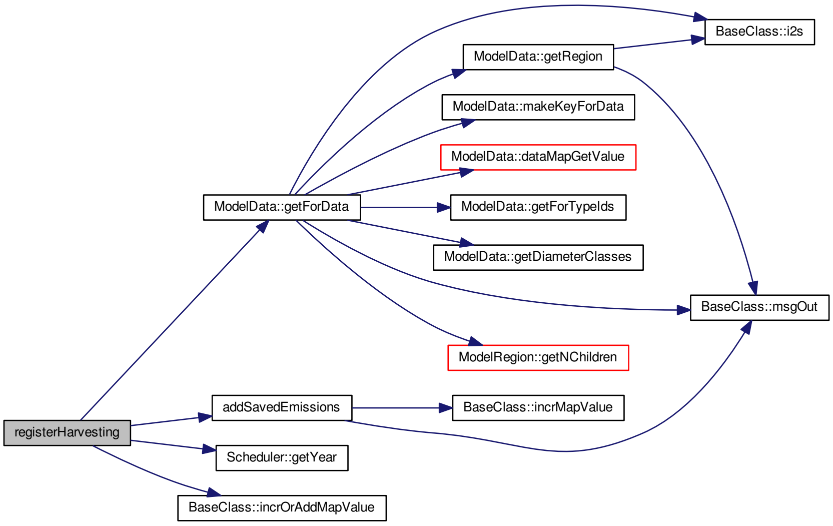
Registers the harvesting of trees increasing the value of cumEmittedForOper.

Definition at line 225 of file Carbon.cpp.

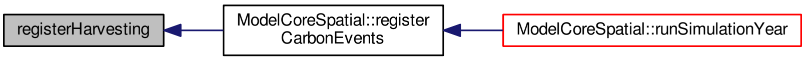
Referenced by ModelCoreSpatial::registerCarbonEvents().

```
00225                                                                                                   {
00226   double convCoeff = MTHREAD->MD->getForData("forOperEmissions",regId,fType,""); //  Kg
       of CO2 emitted per cubic meter of forest operations
00227   // units:
00228   // value: Mm^3
00229   // convCoeff: Kg CO2/m^3 wood
00230   // desidered output: Mt CO2
00231   // ==> I must divide by 1000
00232   addSavedEmissions(-convCoeff*value/1000,regId,EM_FOROP);
00233   // Add the extraBiomass associated to the harvested volumes to the deathBiomassExtra pool
00234   int     year                 = MTHREAD->SCD->getYear();
00235   double extraBiomass_ratio = MTHREAD->MD->getForData("extraBiomass_ratio",regId,fType,"
      ",DATA_NOW);
00236   double newDeathBiomass = value*extraBiomass_ratio;
00237   iiskey key(year,regId,fType);
00238   incrOrAddMapValue(deathBiomassExtra, key, newDeathBiomass);
00239 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


registerHarvesting
ModelCoreSpatial::register CarbonEvents
ModelCoreSpatial::runSimulationYear

4.4.3.21 void registerProducts (const double & *value*, const int & *regId*, const string & *productName*)

Registers the production of a given amount of products, storing it in the products maps. Also increase material substitution.

4.4.3.22 void registerProducts (const double & *value*, const int & *regId*, const string & *productName*)

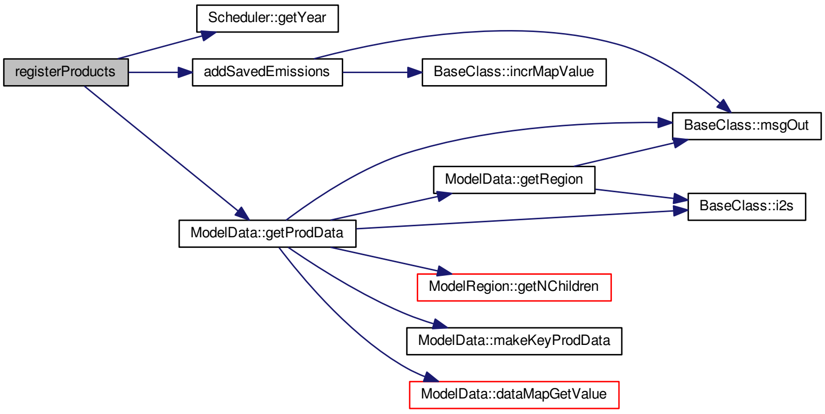
Registers the production of a given amount of products, storing it in the products maps. Also increase material substitution.

Definition at line 257 of file Carbon.cpp.

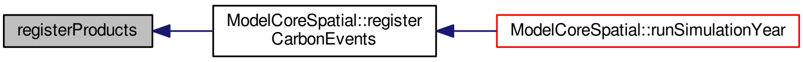
Referenced by ModelCoreSpatial::registerCarbonEvents().

```
00257                                                                                              {
00258   // Registering the CO2 stock embedded in the product...
00259   int year = MTHREAD->SCD->getYear();
00260   iiskey key(year,regId,productName);
00261   pair<iiskey,double> mypair(key,value);
00262   products.insert(mypair);
00263   // registering the substituted CO2 for energy and material..
00264   double subEnergyCoeff   = MTHREAD->MD->getProdData("co2sub_energy",regId,productName,
      DATA_NOW,"");
00265   double subMaterialCoeff = MTHREAD->MD->getProdData("co2sub_material",regId,
      productName,DATA_NOW,"");
00266   // units:
00267   // value: Mm^3
00268   // subEnergyCoeff and subMaterialCoeff:  [kgCO2/m^3 wood]
00269   // desidered output: Mt CO2
00270   // ==> I must divide by 1000
00271   //addSavedEmissions(subEnergyCoeff*value/1000,regId,EM_ENSUB);
00272   addSavedEmissions(subMaterialCoeff*value/1000,regId,EM_MATSUB);
00273 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


registerProducts
ModelCoreSpatial::register CarbonEvents
ModelCoreSpatial::runSimulationYear

4.4.3.23 void registerTransports (const double & *distQ*, const int & *regId*)

Registers the quantities emitted by transport of wood FROM a given region.

4.4.3.24 void registerTransports (const double & *distQ*, const int & *regId*)

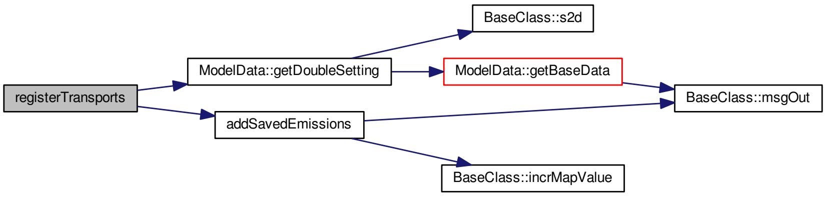
Registers the quantities emitted by transport of wood FROM a given region.

Definition at line 278 of file Carbon.cpp.

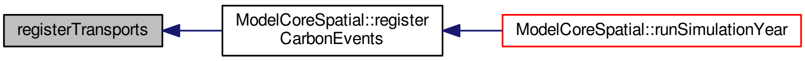
Referenced by ModelCoreSpatial::registerCarbonEvents().

```
00278                                                                {
00279   // units:
00280   // distQ: km*Mm^3
00281   // transportEmissionsCoeff:  [Kg CO2 / (km*m^3) ]
00282   // desidered output: Mt CO2
00283   // ==> I must divide by 1000
00284   double transportEmissionsCoeff = MTHREAD->MD->getDoubleSetting("
      transportEmissionsCoeff");
00285   addSavedEmissions(-transportEmissionsCoeff*distQ/1000,regId,
      EM_FOROP);
00286 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


registerTransports
ModelCoreSpatial::register CarbonEvents
ModelCoreSpatial::runSimulationYear

4.4.4 Member Data Documentation

4.4.4.1 map< int, double > cumEmittedForOper [private]

Map that store emissions for forest operations, including transport, by l2_region [Mt CO2].

Definition at line 78 of file Carbon.h.

Referenced by addSavedEmissions(), getCumSavedEmissions(), and initialiseEmissionCounters().

4.4.4.2 map< int, double > cumSubstitutedEnergy [private]

Map that store the cumulative CO2 substituted for energy consumption, by l2_region [Mt CO2].

Definition at line 76 of file Carbon.h.

Referenced by addSavedEmissions(), getCumSavedEmissions(), and initialiseEmissionCounters().

4.4.4.3 map< int, double > cumSubstitutedMaterial [private]

Map that store the cumulative CO2 substituted using less energivory material, by l2_region [Mt CO2].

Definition at line 77 of file Carbon.h.

Referenced by addSavedEmissions(), getCumSavedEmissions(), and initialiseEmissionCounters().

4.4.4.4 map< iiskey, double > deathBiomassExtra [private]

Map that register the death of biomass by year, l2_region and forest type (non-inventoried biomass: branches, roots..) [Mm^3 wood].

Definition at line 74 of file Carbon.h.

Referenced by getStock(), initialiseDeathBiomassStocks(), registerDeathBiomass(), and registerHarvesting().

4.4.4.5 map< iiskey, double > deathBiomassInventory [private]

Map that register the death of biomass by year, l2_region and forest type (inventoried)[Mm^3 wood].

Definition at line 73 of file Carbon.h.

Referenced by getStock(), initialiseDeathBiomassStocks(), and registerDeathBiomass().

4.4.4.6 map< iiskey, double > products [private]

Map that register the production of a given product by year, l2_region and product [Mm^3 wood].

Definition at line 75 of file Carbon.h.

Referenced by getStock(), HWP_eol2energy(), initialiseProductsStocks(), and registerProducts().

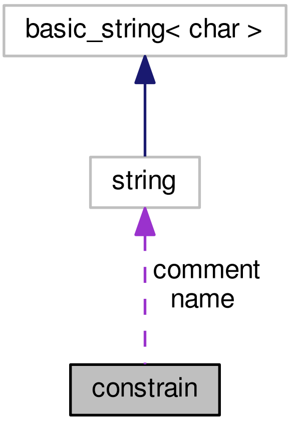
The documentation for this class was generated from the following files:

- /home/lobianco/git/ffsm_pp/src/Carbon.h
- /home/lobianco/git/ffsm_pp/src/MortalityLogBook.h
- /home/lobianco/git/ffsm_pp/src/Carbon.cpp
- /home/lobianco/git/ffsm_pp/src/MortalityLogBook.cpp

4.5 constrain Struct Reference

`#include <Opt.h>`

Collaboration diagram for constrain:



Public Member Functions

- constrain ()

Public Attributes

- string name
- string comment
- int domain
- int direction

4.5.1 Detailed Description

Definition at line 268 of file Opt.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 constrain ( ) `[inline]`

Definition at line 269 of file Opt.h.

```
00269 {comment="";};
```

4.5.3 Member Data Documentation

4.5.3.1 string comment

Definition at line 271 of file Opt.h.

Referenced by Opt::declareConstrains(), and Opt::getConNumber().

4.5.3.2 int direction

Definition at line 273 of file Opt.h.

Referenced by Opt::declareConstrains(), and Opt::getConNumber().

4.5.3.3 int domain

Definition at line 272 of file Opt.h.

Referenced by Opt::declareConstrains(), and Opt::getConNumber().

4.5.3.4 string name

Definition at line 269 of file Opt.h.

Referenced by Opt::declareConstrains(), and Opt::getConNumber().

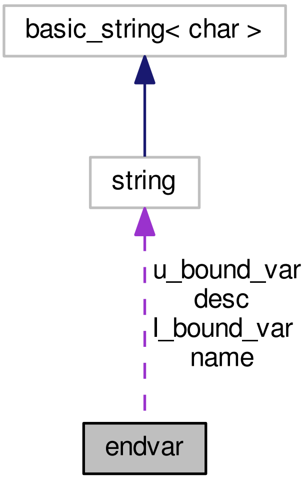
The documentation for this struct was generated from the following file:

- /home/lobianco/git/ffsm_pp/src/Opt.h

4.6 endvar Struct Reference

`#include <Opt.h>`

Collaboration diagram for endvar:



Public Attributes

- string name
- int domain
- string desc

  Description of the variable.
- double l_bound

  A fixed numerical lower bound for all the domain.
- double u_bound

  A fixed numerical upper bound for all the domain.
- string l_bound_var

  A variable giving the lower bound. If present, the value defined in the variable overrides l_bound.
- string u_bound_var

  A variable giving the upper bound. If present, the value defined in the variable overrides u_bound.

4.6.1 Detailed Description

Definition at line 277 of file Opt.h.

4.6.2 Member Data Documentation

4.6.2.1 string desc

Description of the variable.

Definition at line 280 of file Opt.h.

Referenced by Opt::declareVariable().

4.6.2.2 int domain

Definition at line 279 of file Opt.h.

Referenced by Opt::declareVariable(), and Opt::getDetailedBoundByVarAndIndex().

4.6.2.3 double l_bound

A fixed numerical lower bound for all the domain.

Definition at line 281 of file Opt.h.

Referenced by Opt::declareVariable().

4.6.2.4 string l_bound_var

A variable giving the lower bound. If present, the value defined in the variable overrides l_bound.

Definition at line 283 of file Opt.h.

Referenced by Opt::declareVariable(), and Opt::getDetailedBoundByVarAndIndex().

4.6.2.5 string name

Definition at line 278 of file Opt.h.

Referenced by Opt::declareVariable(), and Opt::getDetailedBoundByVarAndIndex().

4.6.2.6 double u_bound

A fixed numerical upper bound for all the domain.

Definition at line 282 of file Opt.h.

Referenced by Opt::declareVariable().

4.6.2.7 string u_bound_var

A variable giving the upper bound. If present, the value defined in the variable overrides u_bound.

Definition at line 284 of file Opt.h.

Referenced by Opt::declareVariable(), and Opt::getDetailedBoundByVarAndIndex().

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

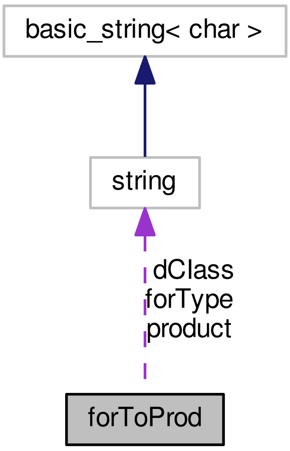
- /home/lobianco/git/ffsm_pp/src/Opt.h

4.7 forToProd Struct Reference

IO production matrix between the forest resources and the primary products (struct)

`#include <ModelData.h>`

Collaboration diagram for forToProd:



Public Attributes

- string product
- string forType
- string dClass
- int maxYears

  The maximum year for a tree collapse that this product can be harvested from. E.g. a 0 value means it can be obtained only from live trees, a 5 years value mean it can be obtained from trees death no longer than 5 years ago.

4.7.1 Detailed Description

IO production matrix between the forest resources and the primary products (struct)

Struct containing the io matrix between the forest resources and the primary products. Not to be confunded with the IO matrix between primary products and secondary products.

Definition at line 271 of file ModelData.h.

4.7.2 Member Data Documentation

4.7.2.1 string dClass

Definition at line 274 of file ModelData.h.

Referenced by ModelData::setDefaultProductResourceMatrixLink(), and ModelData::setScenarioProductResourceMatrixLink().

4.7.2.2 string forType

Definition at line 273 of file ModelData.h.

Referenced by ModelData::setDefaultProductResourceMatrixLink(), and ModelData::setScenarioProductResourceMatrixLink().

4.7.2.3 int maxYears

The maximum year for a tree collapse that this product can be harvested from. E.g. a 0 value means it can be obtained only from live trees, a 5 years value mean it can be obtained from trees death no longer than 5 years ago.

Definition at line 276 of file ModelData.h.

Referenced by ModelData::setDefaultProductResourceMatrixLink().

4.7.2.4 string product

Definition at line 272 of file ModelData.h.

Referenced by ModelData::setDefaultProductResourceMatrixLink(), and ModelData::setScenarioProductResourceMatrixLink().

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

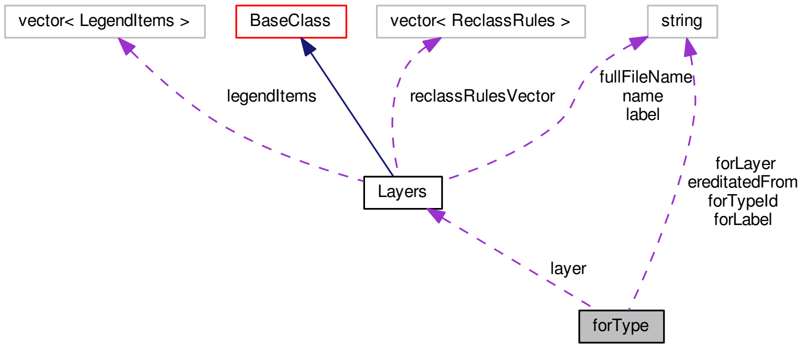
- /home/lobianco/git/ffsm_pp/src/ModelData.h

4.8 forType Struct Reference

Forest types (struct)

`#include <ModelData.h>`

Collaboration diagram for forType:



Public Attributes

- string forTypeId
- string forLabel
- int memType
- string forLayer
- string ereditatedFrom
- Layers ∗ layer

4.8.1 Detailed Description

Forest types (struct)

Struct containing the list of the forest types managed in the model.

memType Parameter to define if this type is used only in initial data reading, then is reclassed and no more used (1) or if it is generated from the reclass

Definition at line 284 of file ModelData.h.

4.8.2 Member Data Documentation

4.8.2.1 string ereditatedFrom

Definition at line 289 of file ModelData.h.

Referenced by Gis::applyForestReclassification(), ModelData::getForTypeChilds_pos(), and ModelData::setForestTypes().

4.8.2.2 string forLabel

Definition at line 286 of file ModelData.h.

Referenced by ModelData::setForestTypes().

4.8.2.3 string forLayer

Definition at line 288 of file ModelData.h.

Referenced by Gis::applyForestReclassification(), ModelCoreSpatial::loadExogenousForestLayers(), and ModelData::setForestTypes().

4.8.2.4 string forTypeId

Definition at line 285 of file ModelData.h.

Referenced by Gis::applyForestReclassification(), and ModelData::setForestTypes().

4.8.2.5 Layers∗ layer

Definition at line 290 of file ModelData.h.

4.8.2.6 int memType

Definition at line 287 of file ModelData.h.

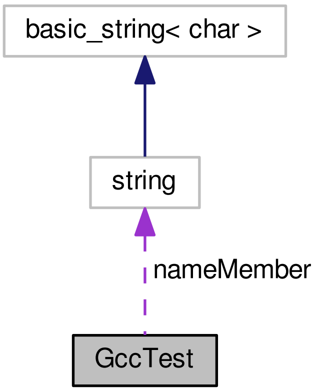
Referenced by ModelCoreSpatial::loadExogenousForestLayers(), and ModelData::setForestTypes().

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

- /home/lobianco/git/ffsm_pp/src/ModelData.h

4.9 GccTest Struct Reference

Collaboration diagram for GccTest:



Public Member Functions

- GccTest (string name_h)
- operator string ()
- operator int ()
- operator vector< int > ()

Public Attributes

- string nameMember

4.9.1 Detailed Description

Definition at line 94 of file Sandbox.cpp.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 GccTest (string *name_h*) `[inline]`

Definition at line 97 of file Sandbox.cpp.

```
00097                      {
00098     nameMember = name_h;
00099   };
```

4.9.3 Member Function Documentation

4.9.3.1 operator int ( ) `[inline]`

Definition at line 111 of file Sandbox.cpp.

```
00112    {
00113     cout << "its \"underload\"\n";
00114       return 42;
00115    }
```

4.9.3.2 operator string ( ) `[inline]`

Definition at line 103 of file Sandbox.cpp.

```
00104    {
00105
00106     cout << "the first function\n";
00107     cout << nameMember << endl;
00108     return "42";
00109    }
```

4.9.3.3 operator vector< int >( ) [inline]

Definition at line 117 of file Sandbox.cpp.

```
{
  cout << "within vector <int>" << endl;
  vector <int> toReturn;
  toReturn.push_back(3);
  toReturn.push_back(4);
  toReturn.push_back(5);
  return toReturn;
}
```

4.9.4 Member Data Documentation

4.9.4.1 string nameMember

Definition at line 99 of file Sandbox.cpp.

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

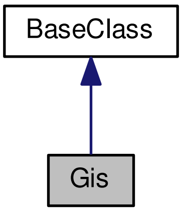
- /home/lobianco/git/ffsm_pp/src/Sandbox.cpp

4.10 Gis Class Reference

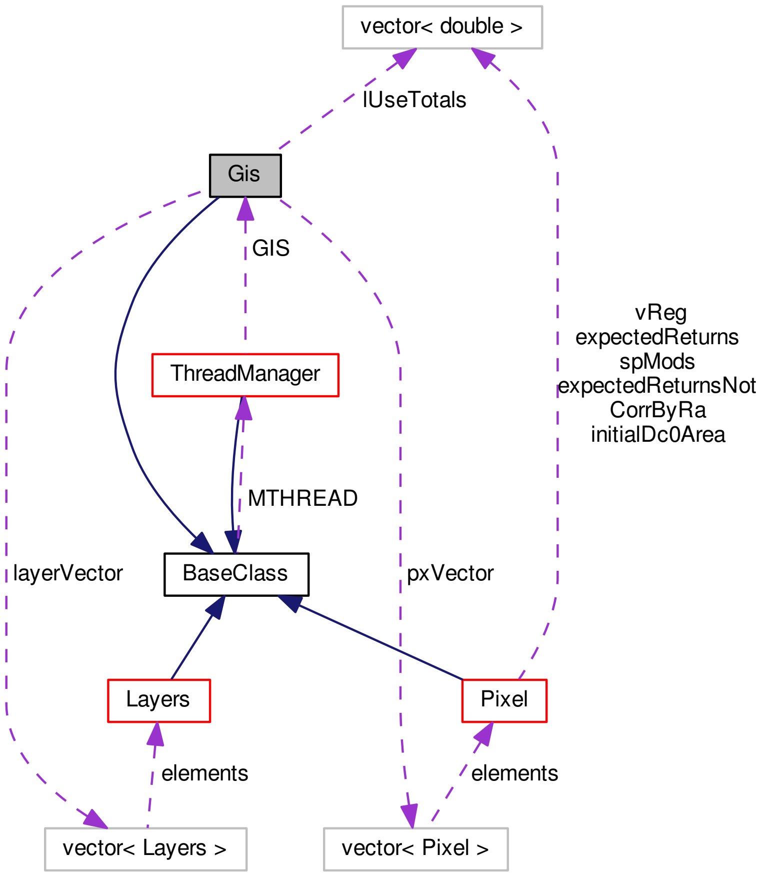
Class to manage the spatial dimension.

`#include <Gis.h>`

Inheritance diagram for Gis:



Collaboration diagram for Gis:



Public Member Functions

- Gis (ThreadManager ∗MTHREAD_h)

  Constructor.
- ∼Gis ()
- void setSpace ()

  Set the initial space environment, including loading data from files.
- void initLayers ()

  Init the layers.
- void initLayersPixelData ()
- void initLayersModelData (const int &year_h=DATA_NOW)
- void applyForestReclassification ()

  Apply the forest reclassification with the rules defined in reclRules sheet.
- void filterSubRegion (string layerName_h)

  If subregion mode is on, this function place noValues on the selected layer for all out-of-region pixels.
- void updateImage (string layerName_h)

  Add one layer to the system.
- void addLayer (string name_h, string label_h, bool isInteger_h, bool dynamicContent_h, string fullFileName_h="", bool display_h=true)

Fill a layer with empty values.

- void resetLayer (string layerName_h)

  Check if a layer with a certain name is loaded in the model. Used e.g. to check if the dtm layer (optional) exist.
- bool layerExist (const string &layerName_h, bool exactMatch=true) const

  Return a pointer to a layer given its name.
- Layers ∗ getLayer (const string &layerName_h)

  Add a legend item to an existing layer.
- void addLegendItem (string name_h, int D_h, string label_h, int rColor_h, int gColor_h, int bColor_h, double minValue_h, double maxValue_h)
- void countItems (const string &layerName_h, const bool &debug=false)

  Count the pixels within each legend item for the selected layer.
- Pixel ∗ getRandomPlotByValue (string layer_h, int layerValue__h)

  Return a pointer to a plot with a specific value for the specified layer.
- vector< Pixel ∗ > getAllPlotsByValue (string layer_h, int layerValue_h, int outputLevel=MSG_WARNING)

  Return the vector (shuffled) of all plots with a specific value for a specified layer. It is also possible to specify the level in case of failure.
- vector< Pixel ∗ > getAllPlotsByValue (string layer_h, vector< int > layerValues_h, int outputLevel=MSG_↩ WARNING)

  Return the vector (shuffled) of all plots with specific values for a specified layer. It is also possible to specify the level in case of failure.
- vector< Pixel ∗ > getAllPlots (int outputLevel=MSG_WARNING)

  Return the vector (shuffled) of all plots. It is also possible to specify the level in case of failure.
- vector< Pixel ∗ > getAllPlotsByRegion (ModelRegion &region_h, bool shuffle=false)

  Return the vector of all plots by a specific region (main region or subregion), optionally shuffled;.
- vector< Pixel ∗ > getAllPlotsByRegion (int regId_h, bool shuffle=false)
- vector< string > getLayerNames ()

  Return a vector of the layer ids (as string)
- vector< Layers ∗ > getLayerPointers ()

  Return a vector of pointers of existing layers.
- void printLayers (string layerName_h="")

  Print the specified layer or all layers (if param layerName_h is missing).
- void printBinMaps (string layerName_h="")

  Save an image in standard png format.
- void printDebugValues (string layerName_h, int min_h=0, int max_h=0)
- double getDistance (const Pixel ∗px1, const Pixel ∗px2)
- int getXNPixels () const
- int getYNPixels () const

  Return the number of pixels on X.
- double getXyNPixels () const

  Return the number of pixels on Y.
- double getHaByPixel () const

  Return the total number of pixels.
- double getNoValue () const
- Pixel ∗ getPixel (int x_h, int y_h)
- Pixel ∗ getPixel (int ID_h)

  Return a pixel pointer from its coordinates.
- double getGeoTopY () const

  Return a pixel pointer from its ID.
- double getGeoBottomY () const
- double getGeoLeftX () const
- double getGeoRightX () const
- double getXMetersByPixel () const

- double getYMetersByPixel () const
- int getSubXL () const
- int getSubXR () const
- int getSubYT () const
- int getSubYB () const
- int sub2realID (int id_h)

  Transform the ID of a pixel in subregion coordinates to the real (and model used) coordinates.
- string pack (const string &parName, const string &forName, const string &dClass, const int &year) const
- void unpack (const string &key, string &parName, string &forName, string &dClass, int &year) const
- void swap (const int &swap_what)

Private Member Functions

- void loadLayersDataFromFile ()

  Load the data of a layer its datafile.
- void applySpatialStochasticValues ()

  Apply stochastic simulation, e.g. regional volume growth s.d. -> tp multipliers.
- void applyStochasticRiskAdversion ()

  Give to each agend a stochastic risk adversion. For now Pixel = Agent.
- void cachePixelValues ()

  For computational reasons cache some values in constant layers directly as properties of the pixel object.

Private Attributes

- vector$<$ Pixel $>$ pxVector

  array of Pixel objects
- vector$<$ Layers $>$ layerVector

  array of Layer objects
- vector$<$ double $>$ lUseTotals

  totals, in ha, of area in the region for each type (cached values)
- int xNPixels

  number of pixels along the X dimension
- int yNPixels

  number of pixels along the Y dimension
- double xyNPixels

  total number of pixels
- double xMetersByPixel

  pixel dimension (meters), X
- double yMetersByPixel

  pixel dimension (meters), Y
- double geoLeftX

  geo-coordinates of the map left border
- double geoTopY

  geo-coordinates of the map upper border
- double geoRightX

  geo-coordinates of the map right border
- double geoBottomY

  geo-coordinates of the map bottom border
- double noValue

  value internally use as novalue (individual layer maps can have other values)

- int subXL

  sub region left X
- int subXR

  sub region right X
- int subYT

  sub region top Y
- int subYB

  sub region bottom Y

Additional Inherited Members

4.10.1 Detailed Description

Class to manage the spatial dimension.

Gis class is responsable to provide all methods for spatial analysis.
It is equipped with two important vectors:

- pxVector contains the array of all pixels on the screen
- layerVector contains the layer objects
  Along the model, IDs of pixels are assigned from left to right, from top to down:
  —>
  /
  —>
  /
  —>

Pixel origin (0,0) on the top left corner is also the system used by the underlying libraries, but put attencion that instead geographical coordinates, if we are on the North emisfere, are increasing along the up-right direction.

Author

Antonello Lobianco

Definition at line 67 of file Gis.h.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Gis (ThreadManager ∗ *MTHREAD_h*)

Constructor.

The constructor of the GIS (unique) instance want:

Parameters

RD_h	Pointer to the manager of the regional data
MTHREAD_h	Pointer to the main thread manager

Definition at line 40 of file Gis.cpp.

```
00040                                {
00041   MTHREAD=MTHREAD_h;
00042 }
```

4.10.2.2 ∼Gis ( )

Definition at line 44 of file Gis.cpp.

```
00044          {
00045 }
```

4.10.3 Member Function Documentation

4.10.3.1 void addLayer (string *name_h,* string *label_h,* bool *isInteger_h,* bool *dynamicContent_h,* string *fullFileName_h* = " ", bool *display_h* = `true`)

Fill a layer with empty values.

Called at init time from initLayers, or during model run-time, this function will add a layer to the system.

Parameters

name_h	ID of the layer (no spaces!)
label_h	layer label
type_h	type of the layer, integer or contiguous
dynamicContent↩_h	if it change during the time (so it needs to be printed each year) or not
fullFilename_h	if the layer has to be read at the beginning, the name of the associated datafile (default="")

It:

- had the layer to the layerVector
- set all pixels with nodata for that specific layer
- let the GUI know we have a new layer

Definition at line 499 of file Gis.cpp.

```
00499
                        {
00500   if(name_h == "forArea_ash"){
00501     bool debug = true;
00502   }
00503   for(uint i=0; i<layerVector.size(); i++){
00504     if (layerVector.at(i).getName() == name_h){
00505       msgOut(MSG_ERROR, "Layer already exist with that name");
00506       return;
00507     }
00508   }
00509   Layers LAYER (MTHREAD, name_h, label_h, isInteger_h, dynamicContent_h, fullFileName_h,
      display_h);
00510   layerVector.push_back(LAYER);
00511
00512   for (uint i=0;i<xyNPixels; i++){
00513     pxVector[i].setValue(name_h,noValue);
00514   }
00515   if(display_h){
00516     MTHREAD->addLayer(name_h,label_h);
00517   }
00518 
00519 }
```

4.10.3.2 void addLegendItem (string *name_h*, int *ID_h*, string *label_h*, int *rColor_h*, int *gColor_h*, int *bColor_h*, double *minValue_h*, double *maxValue_h*)

Search within the layerVector and call addLegendItem(...) to the appropriate one.

Called at init time from initLayers, or during model run-time.

Parameters

name_h	Name of the layer
ID_h	ID of the specific lagend item

See also

Layers::addLegendItem

Definition at line 563 of file Gis.cpp.

```
00563 
                                    {
00564 
00565   for(uint i=0; i<layerVector.size(); i++){
00566     if (layerVector.at(i).getName() == name_h){
00567       layerVector.at(i).addLegendItem(ID_h, label_h, rColor_h, gColor_h, bColor_h, minValue_h,
      maxValue_h);
00568       return;
00569     }
00570   }
00571   msgOut(MSG_ERROR, "Trying to add a legend item to a layer that doesn’t exist.");
00572   return;
00573 }
```

4.10.3.3 void applyForestReclassification ( )

Apply the forest reclassification with the rules defined in reclRules sheet.

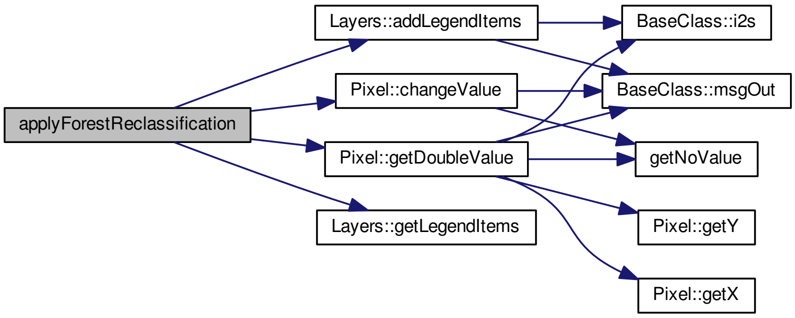
Definition at line 423 of file Gis.cpp.

Referenced by Init::setInitLevel1().

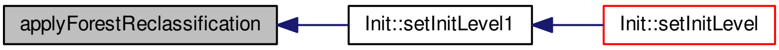
```
00423                                        {
00424 /*per ogni forest type:
00425  - crea i layers delle forest type nuovi
00426  - riempi con zero
00427  - passa le info dal layerr ereditato al nuovo
00428  per ogni pixel
00429  */
00430 
00431   // caching
00432   int nReclassRules = MTHREAD->MD->getNReclRules();
00433   vector <reclRule*> RRs;
00434   for(uint z=0;z<nReclassRules;z++){
00435     RRs.push_back(MTHREAD->MD->getReclRule(z));
```

```
  }



  for (uint i=0;i< MTHREAD->MD->getNForTypes();i++){
    forType* FT = MTHREAD->MD->getForType(i);
    if(!layerExist(FT->forLayer)){
      addLayer(FT->forLayer, "Are layer for forest type "+FT->
forTypeId, false, true);
      resetLayer(FT->forLayer);
      Layers* newLayer = getLayer(FT->forLayer);
      Layers* ereditatedLayer = getLayer(MTHREAD->MD->
getForType(FT->ereditatedFrom)->forLayer);
      newLayer->addLegendItems(ereditatedLayer->getLegendItems());
    }
  }


  for (uint i=0;i< MTHREAD->MD->getNForTypes();i++){
    forType* FT = MTHREAD->MD->getForType(i);
    for(uint j=0;j<xyNPixels;j++){
      Pixel* PX = getPixel(j);
      //int regL1 =  PX->getDoubleValue ("regLev_1");
      int regL2 =  PX->getDoubleValue ("regLev_2");
      double value =  PX->getDoubleValue (FT->forLayer, true);
      for(uint z=0;z<nReclassRules;z++){
        reclRule* RR = RRs[z];
        //if( (RR->regId == regL2 || RR->regId == regL1) && RR->forTypeOut == FT->forTypeId ){ // we found
 a reclassification rule for the region where is located this pixel and that output on the for type we are
 using
        if( RR->regId == regL2  && RR->forTypeOut == FT->
forTypeId ){ // we found a reclassification rule for the region where is located this pixel and
 that output on the for type we are using
          string debugForTypeIn = RR->forTypeIn;
          double inputValue = PX->getDoubleValue(MTHREAD->
MD->getForType(RR->forTypeIn)->forLayer, true);
          double reclassCoeff = RR->coeff;
          value += inputValue * reclassCoeff ;
          // not breaking because we may have more than one input for the same output
        }
      }
      PX->changeValue(FT->forLayer, value, true);
    }
    updateImage(FT->forLayer);
  }
  //countItems("forType_B_HF", true);
  refreshGUI();
  /*Pixel* DP = getPixel(8386);
  msgOut(MSG_DEBUG,"Debug info on plot 8386");
  for (uint i=0;i< MTHREAD->MD->getNForTypes();i++){
    forType* FT = MTHREAD->MD->getForType(i);
    msgOut(MSG_DEBUG,FT->forTypeId+" - "+d2s(DP->getDoubleValue (FT->forLayer)));
  }
  */
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.10.3.4 void applySpatialStochasticValues ( ) [private]

Apply stochastic simulation, e.g. regional volume growth s.d. -> tp multipliers.

Apply all stochastic modifications required by the model at init time. Currently used to change time of passage devending on regional variance with simmetric boundary on the cv I do not change the average, but of course I slighly reduce the stdev. See file monte_carlo_with_multipliers_sample_proof.ods

Definition at line 121 of file Gis.cpp.

```
00121                                  {
00122   // apply regional volume growth st.dev. -> variance to pixel based t.p.
00123   // - cashing value to the pixels
00124   // - apply to the tp layers with change values
00125
00126   if(!MTHREAD->MD->getBoolSetting("usePixelData")) return;
00127
00128   vector <int> regIds2 = MTHREAD->MD->getRegionIds(2);
00129   //ModelRegion* reg = MTHREAD->MD->getRegion(regIds2[i]);
00130   //vector<Pixel*> regPixels = region->getMyPixels();
00131   //double sumc = 0;
00132   //double nc = 0;
00133   for(uint i=0;i<regIds2.size();i++){
00134     ModelRegion* reg = MTHREAD->MD->getRegion(regIds2[i]);
00135     vector <Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(regIds2[i]);
00136     vector <string> fTypes = MTHREAD->MD->getForTypeIds();
00137
00138     // regional variance
00139     if(MTHREAD->MD->getBoolSetting("useSpatialRegionalVariance")){
00140       for(uint j=0; j<fTypes.size(); j++){
00141         double sStDev = MTHREAD->MD->getForData("sStDev",regIds2[i],fTypes[j],""); //
       spatial standard deviation
00142         double agr = MTHREAD->MD->getForData("agr",regIds2[i],fTypes[j],""); // average
       growth
00143         // BUG solved 20141220 To obtain a population with the same avg and st.dev of the original using
       moltipliers, I need to use the cv not the st.dev. !
00144         // tested with excel
00145         normal_distribution<double> d(1,sStDev/agr); // default any how to double
00146         for (uint z=0;z<rpx.size();z++){
00147           double c = d(*MTHREAD->gen);
00148           double c2 = max(0.4,min(1.6,c)); /// with simmetric boundary on the cv I do not change the
       average, but of course I slighly reduce the stdev. See file monte_carlo_with_multipliers_sample_proof.ods
00149           // TO.DO: Convert it to using normSample where instead of a min/max a loop is used to fund
       smaples that are within the bounds
00150           //cout << regIds2[i] << ";" <<sStDev <<";"<< c <<endl
00151           //rpx[z]->correctInputMultiplier("tp_multiplier",fTypes[j],c);
00152           //cout << sStDev/agr << "    " << c2 << endl;
00153           rpx[z]->setSpModifier(c2,j);
00154           //sumc += c;
00155           //nc ++;
00156         }
00157       }
00158     }
00159
00160     // expectation types
00161     double avgExpTypes       = MTHREAD->MD->getDoubleSetting("expType");
00162     double avgExpTypesPrices = MTHREAD->MD->getDoubleSetting("expTypePrices");
00163     double expTypes_cv       = MTHREAD->MD->getDoubleSetting("expType_cv");
00164     double expTypesPrices_cv = MTHREAD->MD->getDoubleSetting("expTypePrices_cv");
00165     if((avgExpTypes<0 || avgExpTypes>1) && avgExpTypes != -1){
00166       msgOut(MSG_CRITICAL_ERROR, "expType parameter must be between 1
       (expectations) and 0 (adaptative) or -1 (fixed).");
00167     }
00168     if(avgExpTypesPrices<0 || avgExpTypesPrices>1){
```

```
00169       msgOut(MSG_CRITICAL_ERROR, "vgExpTypesPrices parameter must be between 1 (expectations) and 0 (adaptative).");
00170     }
00171     //cout << avgExpTypes << "      " << expTypes_cv << endl;
00172 
00173     normal_distribution<double> exp_distr(avgExpTypes,expTypes_cv *avgExpTypes); // works only for double, but default any how to double
00174     normal_distribution<double> expPrices_distr(avgExpTypesPrices,expTypesPrices_cv *avgExpTypesPrices);
00175 
00176     for (uint z=0;z<rpx.size();z++){
00177       if(avgExpTypes == -1){
00178         rpx[z]->expType = -1;
00179       } else {
00180         //double c = exp_distr(*MTHREAD->gen);
00181         //double c2 = max(0.0,min(1.0,c)); /// Bounded [0,1]. With simmetric boundary on the cv I do not change the average, but of course I slighly reduce the stdev. See file monte_carlo_with_multipliers_sample_proof.ods
00182         double c3 = normSample(exp_distr,*MTHREAD->gen,0.0,1.0);
00183         //cout << "Sampled:\t" << c3 <<  endl;
00184         rpx[z]->expType = c3;
00185       }
00186       double cPrice = normSample(expPrices_distr,*MTHREAD->gen,0.0,1.0);
00187       rpx[z]->expTypePrices = cPrice;
00188     }
00189   }
00190 }
```

4.10.3.5 void applyStochasticRiskAdversion ( ) [private]

Give to each agend a stochastic risk adversion. For now Pixel = Agent.

Apply to each agent a random risk-adversion coefficient

For now, 1 pixel = 1 agent, and avg and st.dev. are the same in the model, but eventually this can change

Definition at line 198 of file Gis.cpp.

```
00198                                  {
00199   // apply regional volume growth st.dev. -> variance to pixel based t.p.
00200   // - cashing value to the pixels
00201   // - apply to the tp layers with change values
00202 
00203   if(!MTHREAD->MD->getBoolSetting("usePixelData")) return;
00204 
00205   vector <int> regIds2 = MTHREAD->MD->getRegionIds(2);
00206   bool raEnabled = MTHREAD->MD->getBoolSetting("heterogeneousRiskAversion");
00207   for(uint i=0;i<regIds2.size();i++){
00208     ModelRegion* reg = MTHREAD->MD->getRegion(regIds2[i]);
00209     vector <Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(regIds2[i]);
00210     for (uint z=0;z<rpx.size();z++){
00211       if(raEnabled){
00212         double raStDev = MTHREAD->MD->getDoubleSetting("riskAversionAgentSd");
00213         double avg = MTHREAD->MD->getDoubleSetting("riskAversionAgentAverage");
00214         normal_distribution<double> d(avg,raStDev); // default any how to double
00215         double c = d(*MTHREAD->gen);
00216         rpx[z]->setValue ("ra", c);
00217       } else {
00218         rpx[z]->setValue ("ra", 0.0);
00219       }
00220     }
00221   }
00222 }
```

4.10.3.6 void cachePixelValues ( ) [private]

For computational reasons cache some values in constant layers directly as properties of the pixel object.

Set the avalCoef (availability coefficient) from layer

Definition at line 225 of file Gis.cpp.

```
                     {
  /// Set the avalCoef (availability coefficient) from layer
  if(!MTHREAD->MD->getBoolSetting("usePixelData")) return;

  bool applyAvalCoef = MTHREAD->MD->getBoolSetting("applyAvalCoef");
  vector <int> regIds2 = MTHREAD->MD->getRegionIds(2);

  for(uint i=0;i<regIds2.size();i++){
    ModelRegion* reg = MTHREAD->MD->getRegion(regIds2[i]);
    vector <Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(regIds2[i]);
    for (uint p=0;p<rpx.size();p++){
      if(applyAvalCoef){
        rpx[p]->avalCoef = rpx[p]->getDoubleValue("avalCoef", true);
      }
    }
  }
}
```

4.10.3.7 void countItems (const string & *layerName_h*, const bool & *debug* = `false`)

Count the pixels within each legend item for the selected layer.

Search within the layerVector and call countMyPixels(...) to the appropriate one.

Called at init time from initLayers, or during model run-time.

Parameters

layerName_h	Name of the layer
debug	Print the values on the GUI

See also

Layers::countMyPixels

Definition at line 583 of file Gis.cpp.

```
00583                                                             {
00584 
00585   for(uint i=0; i<layerVector.size(); i++){
00586     if (layerVector.at(i).getName() == layerName_h){
00587       layerVector.at(i).countMyPixels(debug);
00588       return;
00589     }
00590   }
00591   msgOut(MSG_ERROR, "Trying to get statistics (count pixels) of a layer that doesn't exist."
      );
00592   return;
00593 }
```

4.10.3.8 void filterSubRegion (string *layerName_h*)

If subregion mode is on, this function place noValues on the selected layer for all out-of-region pixels.

Update the image behind a layer to the GUI;

This function filter the region, placing noValue on the selected informative layer on pixels that are outside the sub-region.
It was thinked for speedup the development without have to run the whole model for testing each new implementation, but it can used to see what happen in the model when only a subset of the region is analysed.

Definition at line 889 of file Gis.cpp.

```
00889                                       {
00890  subXL = 0;
00891  subYT = 0;
00892  subXR = xNPixels-1;
00893  subYB = yNPixels-1;
00894 }
```

4.10.3.9 vector< **Pixel** ∗ > getAllPlots (int *outputLevel* = **MSG_WARNING**)

Return the vector (shuffled) of all plots. It is also possible to specify the level in case of failure.

Parameters

onlyFreePlots	Flag to get only plots marked as free (d=false)
outputLevel	Level of output in case of failure (no plots available). Default is warning, but if set as MSG_CRITICAL_ERROR it make stop the model.

Definition at line 807 of file Gis.cpp.

```
00807                                   {
00808   vector <Pixel* > candidates;
00809   for (uint i=0;i<pxVector.size();i++){
00810     candidates.push_back(&pxVector.at(i));
00811   }
00812   if (candidates.size()>0){
00813     random_shuffle(candidates.begin(), candidates.end()); // randomize ther elements of the array... cool
       !!! ;-)))
00814   }
00815   else {
00816     msgOut(outputLevel,"We can't find any free plot.");
00817   }
00818   return candidates;
00819 }
```

4.10.3.10 vector< **Pixel** ∗ > getAllPlotsByRegion (**ModelRegion** & *region_h*, bool *shuffle* = `false`)

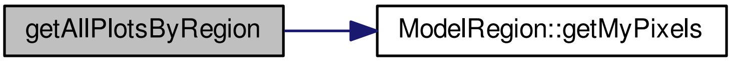
Return the vector of all plots by a specific region (main region or subregion), optionally shuffled;.

Definition at line 823 of file Gis.cpp.

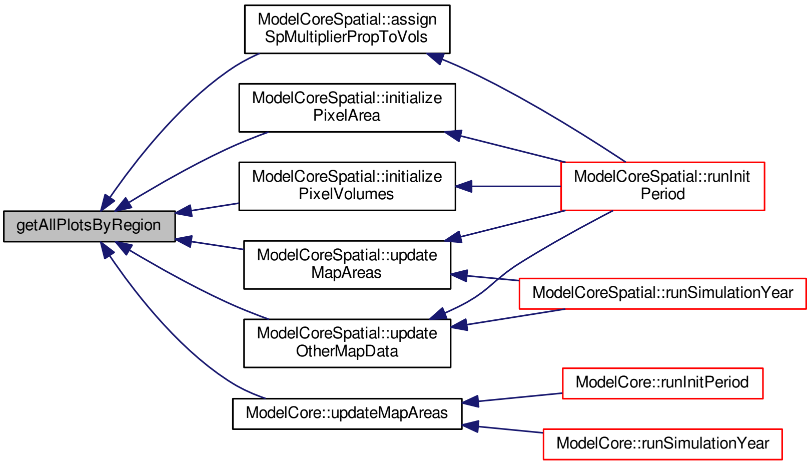
Referenced by ModelCoreSpatial::assignSpMultiplierPropToVols(), ModelCoreSpatial::initializePixelArea(), ModelCoreSpatial::initializePixelVolumes(), ModelCore::updateMapAreas(), ModelCoreSpatial::updateMapAreas(), and ModelCoreSpatial::updateOtherMapData().

```
00823                                                               {
00824   vector <Pixel*> regionalPixels = region_h.getMyPixels();
00825   if(shuffle){
00826     random_shuffle(regionalPixels.begin(), regionalPixels.end()); // randomize the elements of the array.
00827   }
00828   return regionalPixels;
00829 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


ModelCoreSpatial::assign SpMultiplierPropToVols
ModelCoreSpatial::initialize PixelArea
ModelCoreSpatial::initialize PixelVolumes
ModelCoreSpatial::runInit Period
getAllPlotsByRegion
ModelCoreSpatial::update MapAreas
ModelCoreSpatial::runSimulationYear
ModelCoreSpatial::update OtherMapData
ModelCore::runInitPeriod
ModelCore::updateMapAreas
ModelCore::runSimulationYear

4.10.3.11 vector< Pixel * > getAllPlotsByRegion (int *regId_h*, bool *shuffle* = `false`)

Definition at line 832 of file Gis.cpp.

```
00832                                              {
00833     ModelRegion* reg = MTHREAD->MD->getRegion(regId_h);
00834     return getAllPlotsByRegion(*reg,shuffle);
00835 }
```

4.10.3.12 vector< Pixel * > getAllPlotsByValue (string *layer_h*, int *layerValue_h*, int *outputLevel* = MSG_WARNING)

Return the vector (shuffled) of all plots with a specific value for a specified layer. It is also possible to specify the level in case of failure.

Parameters

layer_h	Name of the layer
layerValue←_h	Value we want the plots for
onlyFreePlots	Flag to get only plots marked as free (d=false)
outputLevel	Level of output in case of failure (no plots available). Default is warning, but if set as MSG_CRITICAL_ERROR it make stop the model.

Definition at line 742 of file Gis.cpp.

```
00742                                                                    {
00743   // this would be easier to mantain and cleaned code, but slighly slower:
00744   //vector<int> layerValues;
00745   //layerValues.push_back(layerValue_h);
00746   //return getAllPlotsByValue(layer_h, layerValues, onlyFreePlots, outputLevel);
00747
00748   vector <Pixel* > candidates;
00749   for (uint i=0;i<pxVector.size();i++){
    if(pxVector.at(i).getDoubleValue(layer_h) == layerValue_h){
      candidates.push_back(&pxVector.at(i));
    }
  }

  if (candidates.size()>0){
    random_shuffle(candidates.begin(), candidates.end()); // randomize ther elements of the array... cool !!! ;-)))
  }
  else {
    msgOut(outputLevel,"We can’t find any free plot with "+d2s(layerValue_h)+" value on layer "+layer_h+".");
  }
  return candidates;
}
```

4.10.3.13 vector< Pixel ∗ > getAllPlotsByValue (string *layer_h*, vector< int > *layerValues_h*, int *outputLevel = MSG_WARNING*)

Return the vector (shuffled) of all plots with specific values for a specified layer. It is also possible to specify the level in case of failure.

Parameters

layer_h	Name of the layer
layerValues_h	Values we want the plots for
onlyFreePlots	Flag to get only plots marked as free (d=false)
outputLevel	Level of output in case of failure (no plots available). Default is warning, but if set as MSG_CRITICAL_ERROR it make stop the model.

Definition at line 774 of file Gis.cpp.

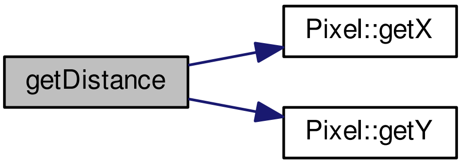
```
                                                                                  {
  vector <Pixel* > candidates;
  string valuesToMatch;
  unsigned int z;

  //string of the required land values to match;
  for (uint j=0;j<layerValues_h.size();j++){
    valuesToMatch = valuesToMatch + " " + i2s(layerValues_h.at(j));
  }

  for (uint i=0;i<pxVector.size();i++){
    z = valuesToMatch.find(d2s(pxVector.at(i).getDoubleValue(layer_h))); // search if in the string of required values is included also the value of the current plot
    if(z!=string::npos){ //z is not at the end of the string, means found!
      candidates.push_back(&pxVector.at(i));
    }
  }

  if (candidates.size()>0){
    random_shuffle(candidates.begin(), candidates.end()); // randomize ther elements of the array... cool !!! ;-)))
  }
  else {
    msgOut(outputLevel,"We can’t find any free plot with the specified values ("+valuesToMatch+") on layer "+layer_h+".");
  }
  return candidates;
}
```

4.10.3.14 double getDistance (const Pixel ∗ *px1*, const Pixel ∗ *px2*)

Definition at line 897 of file Gis.cpp.

```
                                                   {
  return sqrt (
        pow ( (((double)px1->getX()) - ((double)px2->getX()))*xMetersByPixel,2)
        +
        pow ( (((double)px1->getY()) - ((double)px2->getY()))*yMetersByPixel,2)
      );
}
```

Here is the call graph for this function:



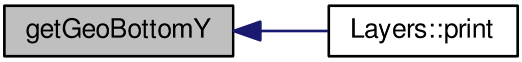
4.10.3.15 double getGeoBottomY ( ) const [inline]

Definition at line 137 of file Gis.h.

Referenced by Layers::print().

```
00137 {return geoBottomY;};
```

Here is the caller graph for this function:



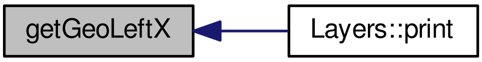
4.10.3.16 double getGeoLeftX ( ) const [inline]

Definition at line 138 of file Gis.h.

Referenced by Layers::print().

```
00138 {return geoLeftX;};
```

Here is the caller graph for this function:



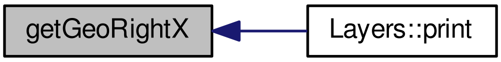
4.10.3.17 double getGeoRightX ( ) const [inline]

Definition at line 139 of file Gis.h.

Referenced by Layers::print().

```
00139 {return geoRightX;};
```

Here is the caller graph for this function:



4.10.3.18 double getGeoTopY ( ) const [inline]

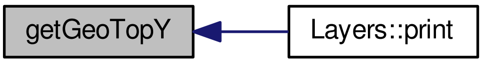
Return a pixel pointer from its ID.

Definition at line 136 of file Gis.h.

Referenced by Layers::print().

```
00136 {return geoTopY;};
```

Here is the caller graph for this function:



4.10.3.19 double getHaByPixel ( ) const [inline]

Return the total number of pixels.

Definition at line 132 of file Gis.h.

```
00132 {return ((xMetersByPixel*yMetersByPixel)/10000) ;};
```

4.10.3.20 Layers ∗ getLayer (const string & *layerName_h*)

Add a legend item to an existing layer.

Init the layers of exogenous data at pixel level (e.g. time of passage) These layers will NOT be read by datafile, but volume for each pixel will be calculated from regional data and area map

Definition at line 413 of file Gis.cpp.

```
00413                                        {
00414   for(uint i=0;i<layerVector.size();i++){
00415     if(layerVector[i].getName() == layerName_h){
00416       return &layerVector[i];
00417     }
00418   }
00419   msgOut(MSG_CRITICAL_ERROR, "Layer "+layerName_h+" not found. Aborting.");
00420 }
```

4.10.3.21 vector< string > getLayerNames ( )

Return a vector of the layer ids (as string)

Definition at line 840 of file Gis.cpp.

```
00840                    {
00841   vector <string> toReturn;
00842   for (uint i=0;i<layerVector.size();i++){
00843     toReturn.push_back(layerVector[i].getName());
00844   }
00845   return toReturn;
00846 }
```

4.10.3.22 vector< Layers ∗ > getLayerPointers ( )

Return a vector of pointers of existing layers.

Definition at line 849 of file Gis.cpp.

```
00849                       {
00850   vector <Layers*> toReturn;
00851   for (uint i=0;i<layerVector.size();i++){
00852     toReturn.push_back(&layerVector[i]);
00853   }
00854   return toReturn;
00855 }
```

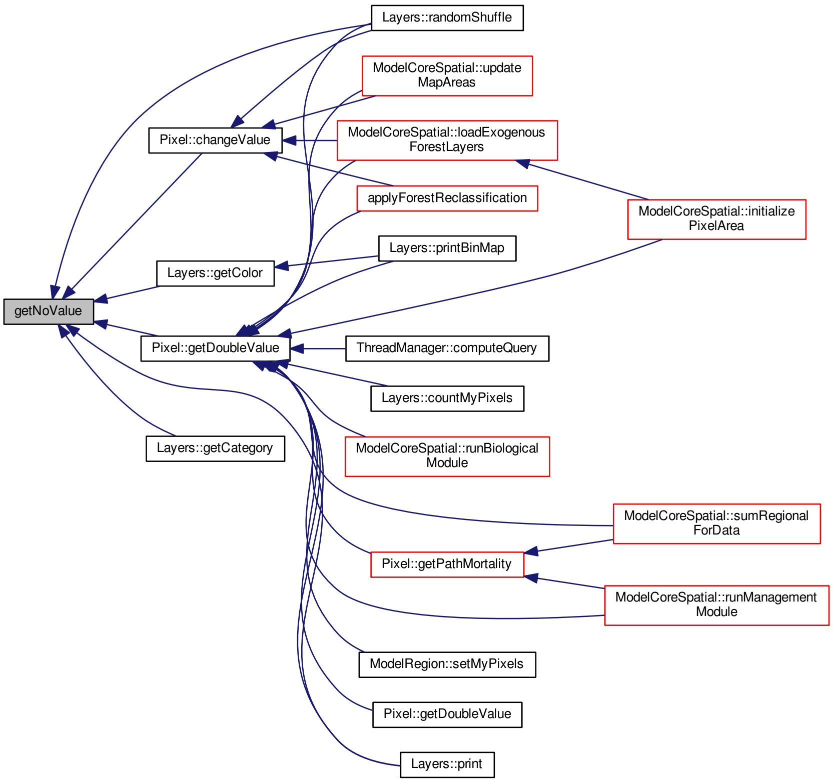
4.10.3.23 double getNoValue ( ) const [inline]

Definition at line 133 of file Gis.h.

Referenced by Pixel::changeValue(), Layers::getCategory(), Layers::getColor(), Pixel::getDoubleValue(), Layers::print(), and Layers::randomShuffle().

```
00133 {return noValue;};
```

Here is the caller graph for this function:


Layers::randomShuffle
ModelCoreSpatial::update MapAreas
Pixel::changeValue
ModelCoreSpatial::loadExogenous ForestLayers
applyForestReclassification
ModelCoreSpatial::initialize PixelArea
Layers::printBinMap
Layers::getColor
getNoValue
Pixel::getDoubleValue
ThreadManager::computeQuery
Layers::countMyPixels
Layers::getCategory
ModelCoreSpatial::runBiological Module
ModelCoreSpatial::sumRegional ForData
Pixel::getPathMortality
ModelCoreSpatial::runManagement Module
ModelRegion::setMyPixels
Pixel::getDoubleValue
Layers::print

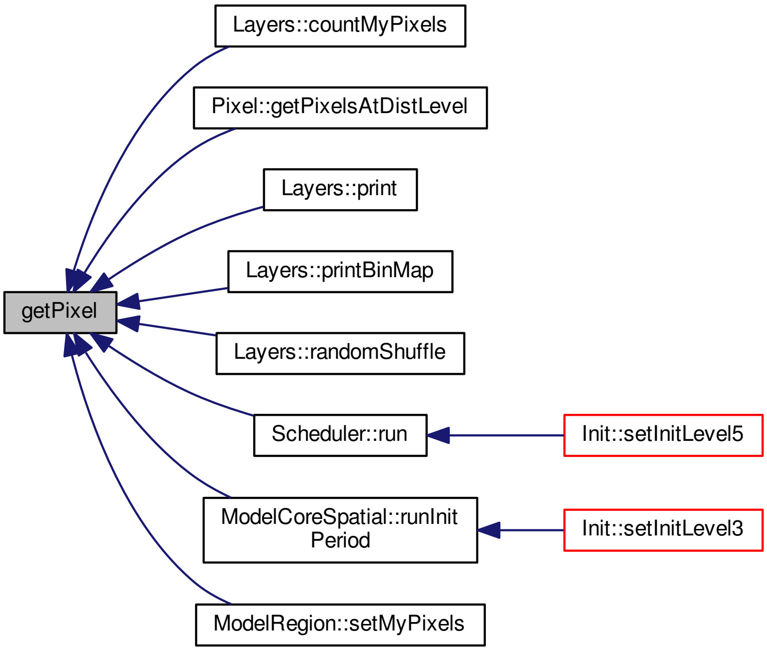
4.10.3.24 Pixel∗ getPixel (int *x_h*, int *y_h*) [inline]

Definition at line 134 of file Gis.h.

Referenced by Layers::countMyPixels(), Pixel::getPixelsAtDistLevel(), Layers::print(), Layers::printBinMap(), Layers::randomShuffle(), Scheduler::run(), ModelCoreSpatial::runInitPeriod(), and ModelRegion::setMyPixels().

```
00134 {return &pxVector.at(x_h+y_h*xNPixels);};  ///< Return a pixel pointer from its coordinates
```

Here is the caller graph for this function:



4.10.3.25 **Pixel**∗ getPixel (int *ID_h*) `[inline]`

Return a pixel pointer from its coordinates.

Definition at line 135 of file Gis.h.

```
00135 {return &pxVector.at(ID_h);};                    ///< Return a pixel pointer from its ID
```

4.10.3.26 **Pixel** ∗ getRandomPlotByValue (string *layer_h*, int *layerValue__h*)

Return a pointer to a plot with a specific value for the specified layer.

Definition at line 714 of file Gis.cpp.

```
00714                                                        {
00715
00716   vector <Pixel* > candidates;
00717   vector <uint> counts;
00718   for(uint i=0;i<pxVector.size();i++) counts.push_back(i);
00719   random_shuffle(counts.begin(), counts.end()); // randomize the elements of the array.
00720
00721   for (uint i=0;i<counts.size();i++){
00722     if(pxVector.at(counts.at(i)).getDoubleValue(layer_h) == layerValue_h ) {
00723         return &pxVector.at(counts.at(i));
00724     }
00725   }
00726
00727   msgOut(MSG_CRITICAL_ERROR,"We can't find any plot with "+
      d2s(layerValue_h)+" value on layer "+layer_h+".");
00728   Pixel* toReturn;
00729   toReturn =0;
00730   return toReturn;
00731 }
```

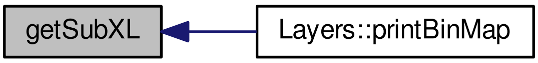
4.10.3.27 int getSubXL ( ) const [inline]

Definition at line 142 of file Gis.h.

Referenced by Layers::printBinMap().

```
00142 {return subXL;};
```

Here is the caller graph for this function:



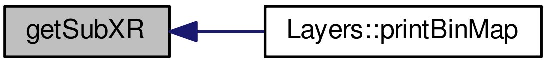
4.10.3.28 int getSubXR ( ) const [inline]

Definition at line 143 of file Gis.h.

Referenced by Layers::printBinMap().

```
00143 {return subXR;};
```

Here is the caller graph for this function:



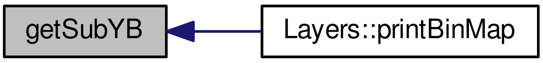
4.10.3.29 int getSubYB ( ) const [inline]

Definition at line 145 of file Gis.h.

Referenced by Layers::printBinMap().

```
00145 {return subYB;};
```

Here is the caller graph for this function:



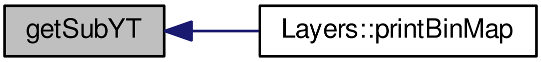
4.10.3.30 int getSubYT ( ) const [inline]

Definition at line 144 of file Gis.h.

Referenced by Layers::printBinMap().

```
00144 {return subYT;};
```

Here is the caller graph for this function:



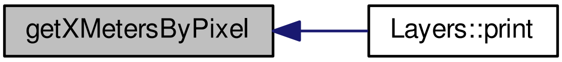
4.10.3.31 double getXMetersByPixel ( ) const [inline]

Definition at line 140 of file Gis.h.

Referenced by Layers::print().

```
00140 {return xMetersByPixel;};
```

Here is the caller graph for this function:



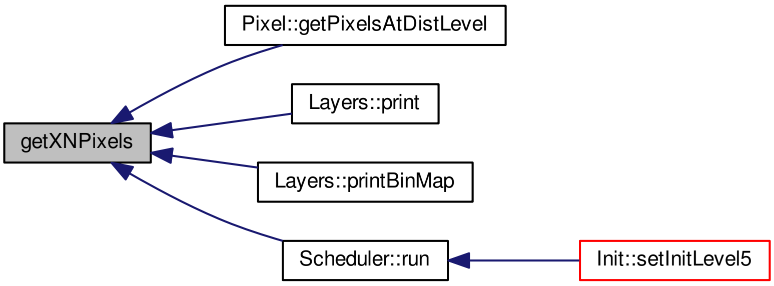
4.10.3.32 int getXNPixels ( ) const [inline]

Definition at line 129 of file Gis.h.

Referenced by Pixel::getPixelsAtDistLevel(), Layers::print(), Layers::printBinMap(), and Scheduler::run().

```
00129 {return xNPixels;};      ///< Return the number of pixels on X
```

Here is the caller graph for this function:



4.10.3.33 double getXyNPixels ( ) const [inline]

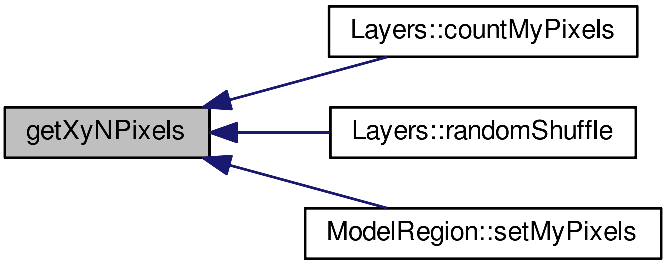
Return the number of pixels on Y.

Definition at line 131 of file Gis.h.

Referenced by Layers::countMyPixels(), Layers::randomShuffle(), and ModelRegion::setMyPixels().

```
00131 {return xyNPixels;};      ///< Return the total number of pixels
```

Here is the caller graph for this function:



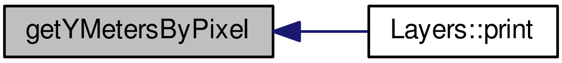
4.10.3.34 double getYMetersByPixel ( ) const [inline]

Definition at line 141 of file Gis.h.

Referenced by Layers::print().

```
00141 {return yMetersByPixel;};
```

Here is the caller graph for this function:



4.10.3.35 int getYNPixels ( ) const [inline]

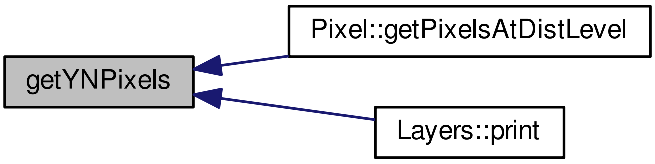
Return the number of pixels on X.

Definition at line 130 of file Gis.h.

Referenced by Pixel::getPixelsAtDistLevel(), and Layers::print().

```
00130 {return yNPixels;};      ///< Return the number of pixels on Y
```

Here is the caller graph for this function:



4.10.3.36 void initLayers ( )

Init the layers.

Called from setSpace(), initLayers() is responsable of:

- load each layer propriety (name, label, datafile..)
- add the layer to the system

  See also

  addLayer

  If the layer is to be read at start-up:
- adding to the layer each legend item (ID, label, min-max values..)

See also

addLegendItem

- [REMOVED, as reclassification rules are in the input ods file now, not in the gis input file] eventually adding to the layer each reclassification rules

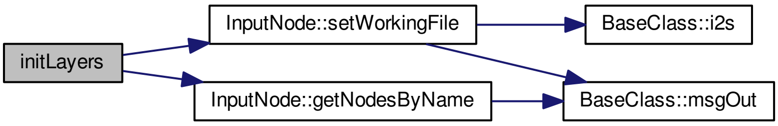
See also

addReclassificationRule

Definition at line 252 of file Gis.cpp.

```
00252                {
00253   // setting layers...
00254   //string filename_complete= MTHREAD->MD->getFilenameByType("gis");
00255   string filename_complete = MTHREAD->getBaseDirectory()+
      MTHREAD->MD->getStringSetting("gisFilename");
00256
00257   InputNode gisDocument;
00258   bool test=gisDocument.setWorkingFile(filename_complete);
00259   if (!test){msgOut(MSG_CRITICAL_ERROR, "Error opening the gis file "+
      filename_complete+".");}
00260   vector<InputNode> layerNodes = gisDocument.getNodesByName("layer");
00261   vector<string> ftIds = MTHREAD->MD->getForTypeIds();
00262   for (uint i=0; i<layerNodes.size();i++){
00263
00264     string nameOrig = layerNodes.at(i).getNodeByName("name").getStringContent();
00265     string labelOrig = layerNodes.at(i).getNodeByName("label").getStringContent();
00266     bool isInteger = layerNodes.at(i).getNodeByName("isInteger").getBoolContent();
00267     bool dynamicContent = layerNodes.at(i).getNodeByName("dynamicContent").getBoolContent();
00268     bool expandByFt = layerNodes.at(i).getNodeByName("expandByFt").getBoolContent();
00269     string readAtStart = layerNodes.at(i).getNodeByName("readAtStart").getStringContent();
00270     if (readAtStart != "true") continue;
00271     string dirName = layerNodes.at(i).getNodeByName("dirName").getStringContent();
00272     string fileName = layerNodes.at(i).getNodeByName("fileName").getStringContent();
00273
00274     // Eventually expanding this input layern in as many layer as forest types exists..
00275     uint endingLoop = expandByFt ? ftIds.size(): 1;
00276     for(uint z=0;z<endingLoop;z++){
00277       string ftExtension= expandByFt ? "_"+ftIds[z]:"";
00278       string labelFtExtension= expandByFt ? " ("+ftIds[z]+")":"";
00279       string name = nameOrig+ftExtension;
00280       string label = labelOrig + labelFtExtension;
00281
00282       string fullFileName = ((dirName == "") || (fileName==""))?"":MTHREAD->
      MD->getBaseDirectory()+dirName+fileName+ftExtension; // TODO: ugly: one would have to put
       mmyfile.grd_broadL_highF
00283       addLayer(name,label,isInteger,dynamicContent,fullFileName);
00284       //legend..
00285       vector<InputNode> legendItemsNodes = layerNodes.at(i).getNodesByName("legendItem");
00286       for (uint j=0; j<legendItemsNodes.size();j++){
00287         int lID = legendItemsNodes.at(j).getIntContent();
00288         string llabel = legendItemsNodes.at(j).getStringAttributeByName("label");
00289         int rColor = legendItemsNodes.at(j).getIntAttributeByName("rColor");
00290         int gColor = legendItemsNodes.at(j).getIntAttributeByName("gColor");
00291         int bColor = legendItemsNodes.at(j).getIntAttributeByName("bColor");
00292         double minValue, maxValue;
00293         if (isInteger){
00294           minValue = ((double)lID);
00295           maxValue = ((double)lID);
00296         }
00297         else {
00298           minValue = legendItemsNodes.at(j).getDoubleAttributeByName("minValue");
00299           maxValue = legendItemsNodes.at(j).getDoubleAttributeByName("maxValue");
00300         }
00301         addLegendItem(name, lID, llabel, rColor, gColor, bColor, minValue, maxValue);
00302       }
00303     }
00304   }
00305   initLayersPixelData();
00306   //initLayersModelData(DATA_INIT); // only the layers relative to the initial years are inserted now. All
       the simulation year layers will be added each year before mainSimulationyear()
00307 }
```

Here is the call graph for this function:



4.10.3.37 void initLayersModelData (const int & *year_h* = DATA_NOW)

4.10.3.38 void initLayersPixelData ( )

Init the layers of exogenous data at pixel level (e.g. time of passage, multipliers, volumes of sp. espl. ft, spread models) These layers will then be read from datafile

Definition at line 313 of file Gis.cpp.

```
00313                          {
00314   if (!MTHREAD->MD->getBoolSetting("usePixelData")){return;}
00315   string dir = MTHREAD->MD->getBaseDirectory()+MTHREAD->
      MD->getStringSetting("spatialDataSubfolder");
00316   string fileExt = MTHREAD->MD->getStringSetting("spatialDataFileExtension");
00317   vector<string> files = vector<string>();
00318   string fullFilename, filename, fullPath;
00319   //string parName, forName, dClass, yearString;
00320   //int year;
00321
00322   MTHREAD->MD->getFilenamesByDir (dir,files, fileExt); // Ugly format. Files is
       the output (reference)
00323
00324   for (unsigned int i = 0;i < files.size();i++) {
00325     fullFilename = files[i];
00326     fullPath = dir+"/"+fullFilename;
00327     filename = fullFilename.substr(0,fullFilename.find_last_of("."));
00328     addLayer(filename,filename,false,false,fullPath,false);
00329   }
00330
00331   // Loading volumes of forest types that are spatially known..
00332   if(MTHREAD->MD->getBoolSetting("useSpExplicitForestTypes")){
00333     string dir2 = MTHREAD->MD->getBaseDirectory()+
      MTHREAD->MD->getStringSetting("spExplicitForTypesInputDir");
00334     string fileExt2 = MTHREAD->MD->getStringSetting("
      spExplicitForTypesFileExtension");
00335     vector<string> files2 = vector<string>();
00336     string fullFilename2, filename2, fullPath2;
00337     MTHREAD->MD->getFilenamesByDir (dir2,files2, fileExt2); // Ugly format. Files
       is the output (reference)
00338     for (unsigned int i = 0;i < files2.size();i++) {
00339       fullFilename2 = files2[i];
00340       fullPath2 = dir2+"/"+fullFilename2;
00341       filename2 = fullFilename2.substr(0,fullFilename2.find_last_of("."));
00342       addLayer(filename2,filename2,false,false,fullPath2,false);
00343     }
00344   }
00345
00346   // Loading pathogens exogenous spread models...
00347   if(MTHREAD->MD->getBoolSetting("usePathogenModule")){
00348     string dir2 = MTHREAD->MD->getBaseDirectory()+
      MTHREAD->MD->getStringSetting("pathogenExogenousSpreadModelFolder");
00349     string fileExt2 = MTHREAD->MD->getStringSetting("
      pathogenExogenousSpreadModelFileExtension");
00350     vector<string> files2 = vector<string>();
00351     string fullFilename2, filename2, fullPath2;
00352     MTHREAD->MD->getFilenamesByDir (dir2,files2, fileExt2); // Ugly format. Files
       is the output (reference)
00353     for (unsigned int i = 0;i < files2.size();i++) {
00354       fullFilename2 = files2[i];
00355       fullPath2 = dir2+"/"+fullFilename2;
00356       filename2 = fullFilename2.substr(0,fullFilename2.find_last_of("."));
00357       addLayer(filename2,filename2,false,false,fullPath2,false);
00358     }
00359   }
00360
00361 }
```

4.10.3.39 bool layerExist (const string & *layerName_h*, bool *exactMatch* = `true`) const

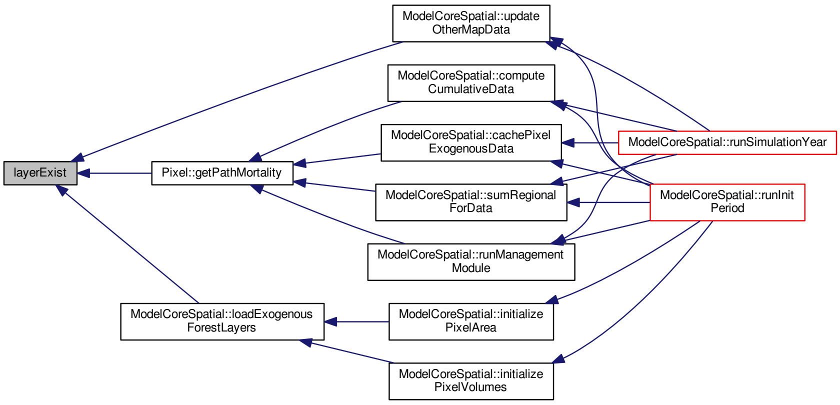
Return a pointer to a layer given its name.

Definition at line 536 of file Gis.cpp.

Referenced by Pixel::getPathMortality(), ModelCoreSpatial::loadExogenousForestLayers(), and ModelCoreSpatial::updateOtherMapData().

```
00536                                                                  {
00537
00538   if(exactMatch){
00539     for(uint i=0; i<layerVector.size(); i++){
00540       if (layerVector.at(i).getName() == layerName_h){
00541         return true;
00542       }
00543     }
00544   } else { // partial matching (stored layer name begin with search parameter)
00545     for(uint i=0; i<layerVector.size(); i++){
00546       if (layerVector.at(i).getName().compare(0, layerName_h.size(),layerName_h )){
00547         return true;
00548       }
00549     }
00550   }
00551
00552   return false;
00553 }
```

Here is the caller graph for this function:


ModelCoreSpatial::update OtherMapData
ModelCoreSpatial::compute CumulativeData
ModelCoreSpatial::cachePixel ExogenousData
ModelCoreSpatial::runSimulationYear
layerExist
Pixel::getPathMortality
ModelCoreSpatial::sumRegional ForData
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::loadExogenous ForestLayers
ModelCoreSpatial::initialize PixelArea
ModelCoreSpatial::initialize PixelVolumes

4.10.3.40 void loadLayersDataFromFile ( ) `[private]`

Load the data of a layer its datafile.

Called at init time from initLayers, this function load the associated datafile to the existing layers (that if exists at this stage are all of type to be loaded at start-up).
This function loop over layerVector and works with GRASS/ASCII (tested) or ARC/ASCII (untested) datasets, assigning to each pixel the readed value to the corresponding layer.
The function also "compose" the initial map with the colors read by the layer (for each specific values) and send the map to the GUI.

NOTE: It uses some Qt functions!!!

See also

Pixel::changeValue
Layers::filterExogenousDataset
Layers::getColor

Definition at line 608 of file Gis.cpp.

```
00608                            {
00609   double localNoValue = noValue;
00610   double inputValue;
00611   double outputValue;
00612   QColor color;
00613 
00614   for(uint i=0;i<layerVector.size();i++){
00615     string layerName =layerVector.at(i).getName();
00616     string fileName=layerVector.at(i).getFilename();
00617     if(fileName == "") continue; // BUGGED !!! 20121017, Antonello. It was "return", so it wasn't reading
 any layers following a layer with no filename
00618     QFile file(fileName.c_str());
00619     if (!file.open(QFile::ReadOnly)) {
00620       cerr << "Cannot open file for reading: "
00621         << qPrintable(file.errorString()) << endl;
00622       msgOut(MSG_ERROR, "Cannot open map file "+fileName+" for reading.");
00623       continue;
00624     }
00625     QTextStream in(&file);
00626     int countRow = 0;
00627     QImage image = QImage(xNPixels, yNPixels, QImage::Format_RGB32);
00628     image.fill(qRgb(255, 255, 255));
00629     while (!in.atEnd()) {
00630       QString line = in.readLine();
00631       QStringList fields = line.split(' ');
00632       if (
00633         (fields.at(0)== "north:" && fields.at(1).toDouble() != geoTopY)
00634         || ((fields.at(0)== "south:" || fields.at(0) == "yllcorner" ) && fields.at(1).toDouble() !=
geoBottomY)
00635         || (fields.at(0)== "east:" && fields.at(1).toDouble() != geoRightX)
00636         || ((fields.at(0)== "west:" || fields.at(0) == "xllcorner" ) && fields.at(1).toDouble() !=
geoLeftX)
00637         || ((fields.at(0)== "rows:" || fields.at(0) == "nrows" ) && fields.at(1).toInt() !=
yNPixels)
00638         || ((fields.at(0)== "cols:" || fields.at(0) == "ncols" ) && fields.at(1).toInt() !=
xNPixels)
00639       )
00640       {
00641         msgOut(MSG_ERROR, "Layer "+layerName+" has different coordinates. Aborting reading."
);
00642         break;
00643       } else if (fields.at(0)== "null:" || fields.at(0) == "NODATA_value" || fields.at(0) == "nodata_value"
 ) {
00644         localNoValue = fields.at(1).toDouble();
00645       } else if (fields.size() > 5) {
00646         for (int countColumn=0;countColumn<xNPixels;countColumn++){
00647           inputValue = fields.at(countColumn).toDouble();
00648           if (inputValue == localNoValue){
00649             outputValue = noValue;
00650             pxVector.at((countRow*xNPixels+countColumn)).changeValue(layerName,outputValue);
00651             QColor nocolor(255,255,255);
00652             color = nocolor;
00653           }
00654           else {
00655             outputValue=layerVector.at(i).filterExogenousDataset(fields.at(countColumn).toDouble
());
00656             pxVector.at((countRow*xNPixels+countColumn)).changeValue(layerName,outputValue);
00657             color = layerVector.at(i).getColor(outputValue);
00658           }
00659           image.setPixel(countColumn,countRow,color.rgb());
00660         }
00661         countRow++;
00662       }
00663     }
00664     if (MTHREAD->MD->getBoolSetting("initialRandomShuffle") ){
00665       layerVector.at(i).randomShuffle();
00666     }
00667     this->filterSubRegion(layerName);
00668     if(layerVector.at(i).getDisplay()){
00669       MTHREAD->updateImage(layerName,image);
00670       //send the image to the gui...
00671       refreshGUI();
00672     }
00673 
00674 
00675   }
00676 }
```

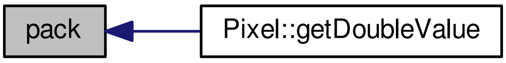
4.10.3.41 string pack (const string & *parName*, const string & *forName*, const string & *dClass*, const int & *year*) const [inline]

Definition at line 148 of file Gis.h.

Referenced by Pixel::getDoubleValue().

```
00148 {return parName+"#"+forName+"#"+dClass+"#"+i2s(year)+"#";};
```

Here is the caller graph for this function:



4.10.3.42 void printBinMaps (string *layerName_h* = "")

Save an image in standard png format.

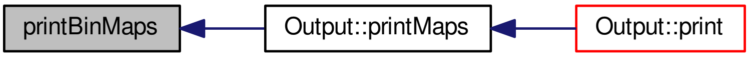
Print debug information (for each pixel in the requested interval, their values on the specified layer)

Definition at line 928 of file Gis.cpp.

Referenced by Output::printMaps().

```
00928                                   {
00929   msgOut(MSG_DEBUG,"Printing the maps as images");
00930   int iteration = MTHREAD->SCD->getIteration(); // are we on the first year of the
       simulation ??
00931   if(layerName_h == ""){
00932     for (uint i=0;i<layerVector.size();i++){
00933       if (!iteration || layerVector[i].getDynamicContent()) {
      layerVector[i].printBinMap();}
00934     }
00935   } else {
00936     for (uint i=0;i<layerVector.size();i++){
00937       if(layerVector[i].getName() == layerName_h){
00938         if (!iteration || layerVector[i].getDynamicContent()) {
      layerVector[i].printBinMap();}
00939         return;
00940       }
00941     }
00942     msgOut(MSG_ERROR, "Layer "+layerName_h+" unknow. No layer printed.");
00943   }
00944 }
```

Here is the caller graph for this function:



4.10.3.43 void printDebugValues (string *layerName_h*, int *min_h* = 0, int *max_h* = 0)

Definition at line 858 of file Gis.cpp.

```
00858                                                              {
00859   int min=min_h;
00860   int max;
00861   int ID, X, Y;
00862   string out;
00863   double value;
00864   //double noValue = MTHREAD->MD->getDoubleSetting("noValue");
00865   if (max_h==0){
00866     max= pxVector.size();
00867   }
00868   else {
00869     max = max_h;
00870   }
00871   msgOut(MSG_DEBUG,"Printing debug information for layer "+layerName_h+".");
00872   for (int i=min;i<max;i++){
00873     value = pxVector.at(i).getDoubleValue(layerName_h);
00874     if (value != noValue){
00875       ID    = i;
00876       X     = pxVector.at(i).getX();
00877       Y     = pxVector.at(i).getY();
00878       out = "Px. "+i2s(ID)+" ("+i2s(X)+","+i2s(Y)+"): "+d2s(value);
00879       msgOut(MSG_DEBUG,out);
00880     }
00881   }
00882 }
```

4.10.3.44 void printLayers (string *layerName_h* = "")

Print the specified layer or all layers (if param layerName_h is missing).

See also

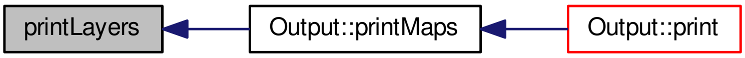
Layers::print()

Definition at line 908 of file Gis.cpp.

Referenced by Output::printMaps().

```
00908                                 {
00909   msgOut(MSG_DEBUG,"Printing the layers");
00910   int iteration = MTHREAD->SCD->getIteration(); // are we on the first year of the
       simulation ??
00911   if(layerName_h == ""){
00912     for (uint i=0;i<layerVector.size();i++){
00913       // not printing if we are in a not-0 iteration and the content of the map doesn't change
00914       if (!iteration || layerVector[i].getDynamicContent())
      layerVector[i].print();
00915     }
00916   } else {
00917     for (uint i=0;i<layerVector.size();i++){
00918       if(layerVector[i].getName() == layerName_h){
00919         if (!iteration || layerVector[i].getDynamicContent())
      layerVector[i].print();
00920         return;
00921       }
00922     }
00923     msgOut(MSG_ERROR, "Layer "+layerName_h+" unknow. No layer printed.");
00924   }
00925 }
```

Here is the caller graph for this function:



4.10.3.45 void resetLayer (string *layerName_h*)

Check if a layer with a certain name is loaded in the model. Used e.g. to check if the dtm layer (optional) exist.

Definition at line 522 of file Gis.cpp.

```
00522                                     {
00523 
00524   for(uint i=0; i<layerVector.size(); i++){
00525     if (layerVector.at(i).getName() == layerName_h){
00526       for (uint i=0;i<xyNPixels; i++){
00527         pxVector.at(i).changeValue(layerName_h,noValue); // bug solved 20071022, Antonello
00528       }
00529       return;
00530     }
00531   }
00532   msgOut(MSG_ERROR, "I could not reset layer "+layerName_h+" as it doesn’t exist!");
00533 }
```

4.10.3.46 void setSpace ( )

Set the initial space environment, including loading data from files.

setSpace is called directly from the init system to setting the space environment in the model.
It is responsable to:

- define map dimensions (from setting files)
- create the pixels
- initialize the layer

  See also

  initLayers
- load the layer data from their fdata-files

  See also

  loadLayersDataFromFile
- tell the GUI that our map will have (x,y) dimensions

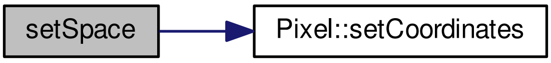
Definition at line 57 of file Gis.cpp.

Referenced by Init::setInitLevel1().

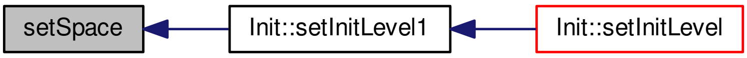
```
00057                 {
00058 
00059 
00060 
00061   msgOut(MSG_INFO,"Creating the space...");
00062 
00063   // init basic settings....
00064   geoTopY = MTHREAD->MD->getDoubleSetting("geoNorthEdge");
00065   geoBottomY = MTHREAD->MD->getDoubleSetting("geoSouthEdge");
00066   geoLeftX = MTHREAD->MD->getDoubleSetting("geoWestEdge");
00067   geoRightX = MTHREAD->MD->getDoubleSetting("geoEastEdge");
00068   xNPixels = MTHREAD->MD->getIntSetting("nCols");
00069   yNPixels = MTHREAD->MD->getIntSetting("nRows");
00070   noValue = MTHREAD->MD->getDoubleSetting("noValue");
00071   xyNPixels = xNPixels * yNPixels;
00072   xMetersByPixel = (geoRightX - geoLeftX)/
      xNPixels;
00073   yMetersByPixel = (geoTopY - geoBottomY)/yNPixels;
00074   MTHREAD->treeViewerChangeGeneralPropertyValue("total plots",
      d2s(getXyNPixels()));
```

```
  MTHREAD->treeViewerChangeGeneralPropertyValue("total land", d2s(xyNPixels*getHaByPixel()));
  // creating pixels...
  for (int i=0;i<yNPixels;i++){
    for (int j=0;j<xNPixels;j++){
      Pixel myPixel(i*xNPixels+j, MTHREAD);
      myPixel.setCoordinates(j,i);
      pxVector.push_back(myPixel);
    }
  }
  initLayers();
  loadLayersDataFromFile();

  // Cashing the pixels owned by each region..
  vector <ModelRegion*> regions = MTHREAD->MD->getAllRegions();
  int nRegions = regions.size();
  for(uint i=0;i<nRegions;i++){
    regions[i]->setMyPixels();
  }

  applySpatialStochasticValues(); // regional variance -> different tp in each pixel trought tp modifiers
  applyStochasticRiskAdversion(); // risk adversion to each pixel
  cachePixelValues(); // For computational reasons cache some values in the constant layers directly as properties of the pixel object

//  //< Print a layer of pixels id..
//  addLayer("pxIds", "idx of the pixels", true, true, "pxIds.grd", true);
//  resetLayer("pxIds");
//  vector<Pixel*> allPixels = getAllPlotsByRegion(11000);
//  for (int i=0;i<allPixels.size();i++){
//    int pxId= allPixels[i]->getID();
//    allPixels[i]->changeValue ("pxIds", pxId);
//  }
//  printLayers("pxIds");


  MTHREAD->fitInWindow(); // tell the gui to fit the map to the widget
//  countItems("landUse",false); // count the various records assigned to each legendItem. Do not print debug infos
  return;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.10.3.47 int sub2realID (int *id_h*)

Transform the ID of a pixel in subregion coordinates to the real (and model used) coordinates.

Definition at line 947 of file Gis.cpp.

```
00947                        {
00948   // IMPORTANT: this function is called at refreshGUI() times, so if there are output messages, call them
       with the option to NOT refresh the gui, otherwise we go to an infinite loop...
00949   return id_h;
00950 }
```

4.10.3.48 void swap (const int & *swap_what*)

Definition at line 970 of file Gis.cpp.

```
00970                              {
00971
00972   for(uint i=0;i<pxVector.size();i++) {
00973     pxVector[i].swap(swap_what);
00974   }
00975
00976 }
```

4.10.3.49 void unpack (const string & *key,* string & *parName,* string & *forName,* string & *dClass,* int & *year*) const

Definition at line 953 of file Gis.cpp.

```
00953                                                                                                            {
00954   int parNameDelimiter = key.find("#",0);
00955   int forNameDelimiter = key.find("#",parNameDelimiter+1);
00956   int dClassDelimiter = key.find("#",forNameDelimiter+1);
00957   int yearDelimiter = key.find("#",dClassDelimiter+1);
00958   if (yearDelimiter == string::npos){
00959     msgOut(MSG_CRITICAL_ERROR, "Error in unpacking the key for the layer.");
00960   }
00961   parName.assign(key,0,parNameDelimiter);
00962   forName.assign(key,parNameDelimiter+1,forNameDelimiter-parNameDelimiter-1);
00963   dClass.assign(key,forNameDelimiter+1,dClassDelimiter-forNameDelimiter-1);
00964   string yearString="";
00965   yearString.assign(key,dClassDelimiter+1,yearDelimiter-dClassDelimiter-1);
00966   year = s2i(yearString);
00967 }
```

4.10.3.50 void updateImage (string *layerName_h*)

Add one layer to the system.

Update an ALREADY EXISTING image and send the updated image to the GUI.
It is used instead of updating the individual pixels that is much more time consuming than change the individual pixels value and then upgrade the image as a whole.

Parameters

layername↩ _h	Layer from where get the image data

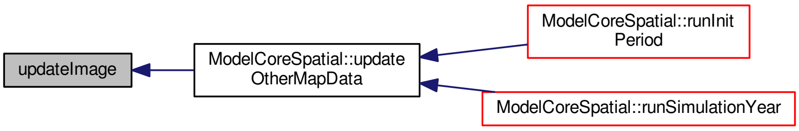
Definition at line 684 of file Gis.cpp.

Referenced by ModelCoreSpatial::updateOtherMapData().

```
00684                                    {
00685   msgOut (1, "Update image "+layerName_h+"...");
00686
00687   // sub{X,Y}{R,L,T,B} refer to the subregion coordinates, but when this is not active they coincide with
```

```
 the whole region
  QImage image = QImage(subXR-subXL+1, subYB-subYT+1, QImage::Format_RGB32);

  image.fill(qRgb(255, 255, 255));
  int layerIndex=-1;
  for (uint i=0;i<layerVector.size();i++){
    if (layerVector.at(i).getName() == layerName_h){
      layerIndex=i;
      break;
    }
  }
  if (layerIndex <0) {
    msgOut(MSG_CRITICAL_ERROR, "Layer not found in Gis::updateImage()");
  }

  for (int countRow=subYT;countRow<subYB;countRow++){
    for (int countColumn=subXL;countColumn<subXR;countColumn++){
      double value = pxVector.at((countRow*xNPixels+countColumn)).getDoubleValue(
layerName_h);
      QColor color = layerVector.at(layerIndex).getColor(value);
      image.setPixel(countColumn-subXL,countRow-subYT,color.rgb());
    }
  }
  MTHREAD->updateImage(layerName_h,image);
  refreshGUI();
}
```

Here is the caller graph for this function:


updateImage
ModelCoreSpatial::update OtherMapData
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

4.10.4 Member Data Documentation

4.10.4.1 double geoBottomY [private]

geo-coordinates of the map bottom border

Definition at line 169 of file Gis.h.

4.10.4.2 double geoLeftX [private]

geo-coordinates of the map left border

Definition at line 166 of file Gis.h.

4.10.4.3 double geoRightX [private]

geo-coordinates of the map right border

Definition at line 168 of file Gis.h.

4.10.4.4 double geoTopY [private]

geo-coordinates of the map upper border

Definition at line 167 of file Gis.h.

4.10.4.5 vector<**Layers**> layerVector `[private]`

array of Layer objects

Definition at line 159 of file Gis.h.

4.10.4.6 vector<double> lUseTotals `[private]`

totals, in ha, of area in the region for each type (cached values)

Definition at line 160 of file Gis.h.

4.10.4.7 double noValue `[private]`

value internally use as novalue (individual layer maps can have other values)

Definition at line 170 of file Gis.h.

4.10.4.8 vector<**Pixel**> pxVector `[private]`

array of Pixel objects

Definition at line 158 of file Gis.h.

4.10.4.9 int subXL `[private]`

sub region left X

Definition at line 171 of file Gis.h.

4.10.4.10 int subXR `[private]`

sub region right X

Definition at line 172 of file Gis.h.

4.10.4.11 int subYB `[private]`

sub region bottom Y

Definition at line 174 of file Gis.h.

4.10.4.12 int subYT `[private]`

sub region top Y

Definition at line 173 of file Gis.h.

4.10.4.13 double xMetersByPixel `[private]`

pixel dimension (meters), X

Definition at line 164 of file Gis.h.

4.10.4.14 int xNPixels `[private]`

number of pixels along the X dimension

Definition at line 161 of file Gis.h.

4.10.4.15 double xyNPixels `[private]`

total number of pixels

Definition at line 163 of file Gis.h.

4.10.4.16 double yMetersByPixel `[private]`

pixel dimension (meters), Y

Definition at line 165 of file Gis.h.

4.10.4.17 int yNPixels `[private]`

number of pixels along the Y dimension

Definition at line 162 of file Gis.h.

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

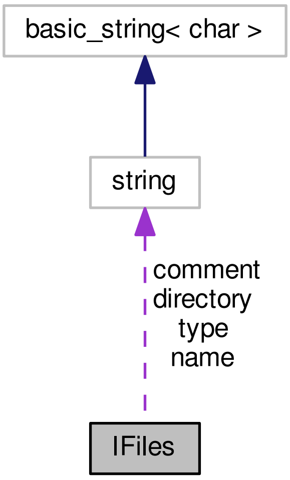
- /home/lobianco/git/ffsm_pp/src/Gis.h
- /home/lobianco/git/ffsm_pp/src/Gis.cpp

4.11 IFiles Struct Reference

Input files (struct)

`#include <ModelData.h>`

Collaboration diagram for IFiles:



Public Attributes

- string directory
- string type
- string name
- string comment

4.11.1 Detailed Description

Input files (struct)

Very short struct containing the input files used (one istance==one file).
A copy of each Istances is saved on vector iFilesVector in class ModelData.
iFiles are defined in the main config file and parsed subsequently.

Author

Antonello Lobianco

Definition at line 247 of file ModelData.h.

4.11.2 Member Data Documentation

4.11.2.1 string comment

Definition at line 251 of file ModelData.h.

4.11.2.2 string directory

Definition at line 248 of file ModelData.h.

4.11.2.3 string name

Definition at line 250 of file ModelData.h.

4.11.2.4 string type

Definition at line 249 of file ModelData.h.

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

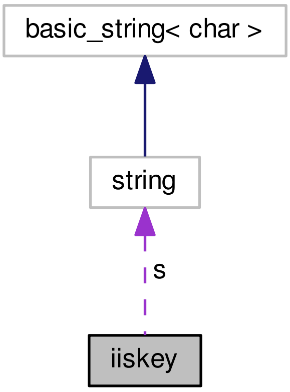
- /home/lobianco/git/ffsm_pp/src/ModelData.h

4.12 iiskey Class Reference

Class to provide a simple integer-integer-string key in std maps.

`#include <BaseClass.h>`

Collaboration diagram for iiskey:



Public Member Functions

- iiskey ()

  iiskey class (note the double ii) ///
- iiskey (int i_h, int i2_h, string s_h)
- ∼iiskey ()
- bool operator== (const iiskey &op2) const
- bool operator!= (const iiskey &op2) const
- bool operator< (const iiskey &op2) const
- bool operator> (const iiskey &op2) const
- bool operator<= (const iiskey &op2) const
- bool operator>= (const iiskey &op2) const

Public Attributes

- int i
- int i2
- string s

4.12.1 Detailed Description

Class to provide a simple integer-integer-string key in std maps.

Definition at line 192 of file BaseClass.h.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 iiskey ( )

iiskey class (note the double ii) ///

Definition at line 469 of file BaseClass.cpp.

```
00469                 {
00470   i = 0;
00471   i2 = 0;
00472   s = "";
00473 }
```

4.12.2.2 iiskey (int *i_h*, int *i2_h*, string *s_h*)

Definition at line 474 of file BaseClass.cpp.

```
00474                                               {
00475   i  = i_h;
00476   i2 = i2_h;
00477   s  = s_h;
00478 }
```

4.12.2.3 ∼iiskey ( )

Definition at line 480 of file BaseClass.cpp.

```
00480                  {
00481
00482 }
```

4.12.3 Member Function Documentation

4.12.3.1 bool operator!= (const iiskey & *op2*) const

Definition at line 493 of file BaseClass.cpp.

```
00493                                                {
00494   if(op2.i == i && op2.i2 == i2 && op2.s == s){
00495     return false;
00496   }
00497   return true;
00498 }
```

4.12.3.2 bool operator< (const iiskey & *op2*) const

Definition at line 501 of file BaseClass.cpp.

```
00501                                               {
00502   if (i < op2.i ) {return true;}
00503   if (i == op2.i) {
00504     if (i2 < op2.i2 ) {return true;}
00505     if (i2 == op2.i2){
00506       if (s < op2.s) {return true;}
00507     }
00508   }
00509   return false;
00510 }
```

4.12.3.3 bool operator<= (const iiskey & *op2*) const

Definition at line 525 of file BaseClass.cpp.

```
00525                                           {
00526   if (i < op2.i ) {return true;}
00527   if (i == op2.i) {
00528     if (i2 < op2.i2 ) {return true;}
00529     if (i2 == op2.i2){
00530       if (s <= op2.s) {return true;}
00531     }
00532   }
00533   return false;
00534 }
```

4.12.3.4 bool operator== (const iiskey & *op2*) const

Definition at line 485 of file BaseClass.cpp.

```
00485                                           {
00486   if(op2.i == i && op2.i2 == i2 && op2.s == s){
00487     return true;
00488   }
00489   return false;
00490 }
```

4.12.3.5 bool operator> (const iiskey & *op2*) const

Definition at line 513 of file BaseClass.cpp.

```
00513                                          {
00514   if (i > op2.i ) {return true;}
00515   if (i == op2.i) {
00516     if (i2 > op2.i2 ) {return true;}
00517     if (i2 == op2.i2){
00518       if (s > op2.s) {return true;}
00519     }
00520   }
00521   return false;
00522 }
```

4.12.3.6 bool operator>= (const iiskey & *op2*) const

Definition at line 537 of file BaseClass.cpp.

```
00537                                           {
00538   if (i > op2.i ) {return true;}
00539   if (i == op2.i) {
00540     if (i2 > op2.i2 ) {return true;}
00541     if (i2 == op2.i2){
00542       if (s >= op2.s) {return true;}
00543     }
00544   }
00545   return false;
00546 }
```

4.12.4 Member Data Documentation

4.12.4.1 int i

Definition at line 203 of file BaseClass.h.

Referenced by operator!=(), operator<(), operator<=(), operator==(), operator>(), and operator>=().

4.12.4.2 int i2

Definition at line 204 of file BaseClass.h.

Referenced by operator!=(), operator<(), operator<=(), operator==(), operator>(), and operator>=().

4.12.4.3 string s

Definition at line 205 of file BaseClass.h.

Referenced by operator!=(), operator<(), operator<=(), operator==(), operator>(), and operator>=().

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

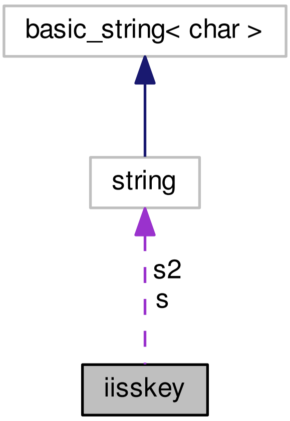
- /home/lobianco/git/ffsm_pp/src/BaseClass.h
- /home/lobianco/git/ffsm_pp/src/BaseClass.cpp

4.13 iisskey Class Reference

Class to provide a simple integer-integer-string-string key in std maps.

`#include <BaseClass.h>`

Collaboration diagram for iisskey:



Public Member Functions

- iisskey ()

  iisskey class (note the double ii and double ss) ///
- iisskey (int i_h, int i2_h, string s_h, string s2_h)
- ∼iisskey ()
- bool filter (const iisskey &key_h) const
- string print () const
- bool operator== (const iisskey &op2) const
- bool operator!= (const iisskey &op2) const
- bool operator< (const iisskey &op2) const
- bool operator> (const iisskey &op2) const
- bool operator<= (const iisskey &op2) const
- bool operator>= (const iisskey &op2) const

Public Attributes

- int i
- int i2
- string s
- string s2

4.13.1 Detailed Description

Class to provide a simple integer-integer-string-string key in std maps.

Definition at line 210 of file BaseClass.h.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 iisskey ( )

iisskey class (note the double ii and double ss) ///

Definition at line 549 of file BaseClass.cpp.

```
00549                   {
00550   i = 0;
00551   i2 = 0;
00552   s = "";
00553   s2= "";
00554 }
```

4.13.2.2 iisskey (int *i_h,* int *i2_h,* string *s_h,* string *s2_h*)

Definition at line 555 of file BaseClass.cpp.

```
00555                                                     {
00556   i  = i_h;
00557   i2 = i2_h;
00558   s  = s_h;
00559   s2 = s2_h;
00560 }
```

4.13.2.3 ∼iisskey ( )

Definition at line 562 of file BaseClass.cpp.

```
00562                    {
00563
00564 }
```

4.13.3 Member Function Documentation

4.13.3.1 bool filter (const iisskey & *key_h*) const

Definition at line 643 of file BaseClass.cpp.

```
00643                                           {
00644   if( (key_h.i == NULL  || key_h.i==i)    &&
00645       (key_h.i2 == NULL || key_h.i2==i2)  &&
00646       (key_h.s == ""  || key_h.s==s)    &&
00647       (key_h.s2 == "" || key_h.s2==s2)     ) return true;
00648   return false;
00649 }
```

4.13.3.2 bool operator!= (const iisskey & *op2*) const

Definition at line 575 of file BaseClass.cpp.

```
00575                                               {
00576   if(op2.i == i && op2.i2 == i2 && op2.s == s && op2.s2 == s2){
00577     return false;
00578   }
00579   return true;
00580 }
```

4.13.3.3 bool operator< (const iisskey & *op2*) const

Definition at line 583 of file BaseClass.cpp.

```
00583                                              {
00584   if (i < op2.i ) {return true;}
00585   if (i == op2.i) {
00586     if (i2 < op2.i2 ) {return true;}
00587     if (i2 == op2.i2){
00588       if (s < op2.s) {return true;}
00589       if (s == op2.s){
00590         if (s2 < op2.s2) {return true;}
00591       }
00592     }
00593   }
00594   return false;
00595 }
```

4.13.3.4 bool operator<= (const iisskey & *op2*) const

Definition at line 613 of file BaseClass.cpp.

```
00613                                               {
00614   if (i < op2.i ) {return true;}
00615   if (i == op2.i) {
00616     if (i2 < op2.i2 ) {return true;}
00617     if (i2 == op2.i2){
00618       if (s < op2.s) {return true;}
00619       if (s == op2.s){
00620         if (s2 <= op2.s2) {return true;}
00621       }
00622     }
00623   }
00624   return false;
00625 }
```

4.13.3.5 bool operator== (const iisskey & *op2*) const

Definition at line 567 of file BaseClass.cpp.

```
00567                                                {
00568   if(op2.i == i && op2.i2 == i2 && op2.s == s && op2.s2 == s2){
00569     return true;
00570   }
00571   return false;
00572 }
```

4.13.3.6 bool operator> (const iisskey & *op2*) const

Definition at line 598 of file BaseClass.cpp.

```
00598                                               {
00599   if (i > op2.i ) {return true;}
00600   if (i == op2.i) {
00601     if (i2 > op2.i2 ) {return true;}
00602     if (i2 == op2.i2){
00603       if (s > op2.s) {return true;}
00604       if (s == op2.s){
00605         if (s2 > op2.s2) {return true;}
00606       }
00607     }
00608   }
00609   return false;
00610 }
```

4.13.3.7 bool operator>= (const iisskey & *op2*) const

Definition at line 628 of file BaseClass.cpp.

```
00628                                                {
00629   if (i > op2.i ) {return true;}
00630   if (i == op2.i) {
00631     if (i2 > op2.i2 ) {return true;}
00632     if (i2 == op2.i2){
00633       if (s > op2.s) {return true;}
00634       if (s == op2.s){
00635         if (s2 >= op2.s2) {return true;}
00636       }
00637     }
00638   }
00639   return false;
00640 }
```

4.13.3.8 string print ( ) const

Definition at line 652 of file BaseClass.cpp.

```
00652                         {
00653     char  outChar1[24];
00654     char  outChar2[24];
00655     snprintf ( outChar1, sizeof(outChar1), "%d", i);
00656     snprintf ( outChar2, sizeof(outChar2), "%d", i2);
00657     return string(outChar1)+’\t’+string(outChar2)+’\t’+s+’\t’+s2;
00658 
00659 }
```

4.13.4 Member Data Documentation

4.13.4.1 int i

Definition at line 223 of file BaseClass.h.

Referenced by filter(), operator!=(), operator<(), operator<=(), operator==(), operator>(), and operator>=().

4.13.4.2 int i2

Definition at line 224 of file BaseClass.h.

Referenced by filter(), operator!=(), operator<(), operator<=(), operator==(), operator>(), and operator>=().

4.13.4.3 string s

Definition at line 225 of file BaseClass.h.

Referenced by filter(), operator!=(), operator<(), operator<=(), operator==(), operator>(), and operator>=().

4.13.4.4 string s2

Definition at line 226 of file BaseClass.h.

Referenced by filter(), operator!=(), operator<(), operator<=(), operator==(), operator>(), and operator>=().

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

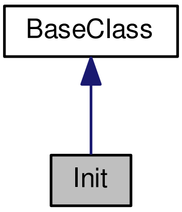
- /home/lobianco/git/ffsm_pp/src/BaseClass.h
- /home/lobianco/git/ffsm_pp/src/BaseClass.cpp

4.14 Init Class Reference

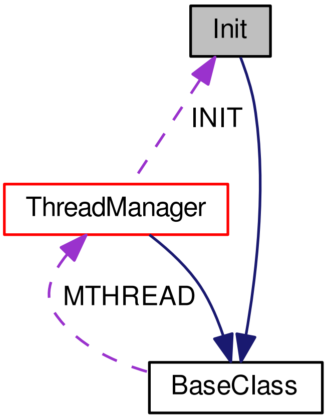
Init the environment, the objects and the agents of the model

`#include <Init.h>`

Inheritance diagram for Init:



Collaboration diagram for Init:



Public Member Functions

- Init (ThreadManager ∗MTHREAD_h)
- ∼Init ()
- void setInitLevel (int level_h)

  Wrapper to the correct setInitLevelX()
- void setInitLevel0 ()

  Unused, reserver for future use.
- void setInitLevel1 ()

  Setting up the space, the model objects and the agents (definitions only)
- void setInitLevel2 ()

  Unused, reserver for future use.
- void setInitLevel3 ()

  Linking object to agents and assigning space proprieties to objects and agents.
- void setInitLevel4 ()

  Unused, reserver for future use.
- void setInitLevel5 ()

  Simulation start.
- void setInitLevel6 ()

  End of simulation (e.g. print summary statistics)
- int getInitState ()

Private Attributes

- int InitState

  One of the 7 possible init states (0..6)
- struct tm ∗ current
- time_t now

Additional Inherited Members

4.14.1 Detailed Description

Init the environment, the objects and the agents of the model

The Init class is responsable to ask to the various objects to Init themself, in a 7-steps procedures.
The basic idea is to first init the environment: options, settings and space.
Then objects and agents are mould up, objects are assigned to agents and finally agents and objects are collocated in the space.

Author

Antonello Lobianco

Definition at line 45 of file Init.h.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 Init (**ThreadManager** ∗ *MTHREAD_h*)

Definition at line 37 of file Init.cpp.

```
00037                                    {
00038   MTHREAD=MTHREAD_h;
00039   InitState=0;
00040 }
```

4.14.2.2 ∼Init ( )

Definition at line 42 of file Init.cpp.

```
00043 {
00044 }
```

4.14.3 Member Function Documentation

4.14.3.1 int getInitState ( ) `[inline]`

Definition at line 67 of file Init.h.

```
00067 {return InitState;};
```

4.14.3.2 void setInitLevel (int *level_h*)

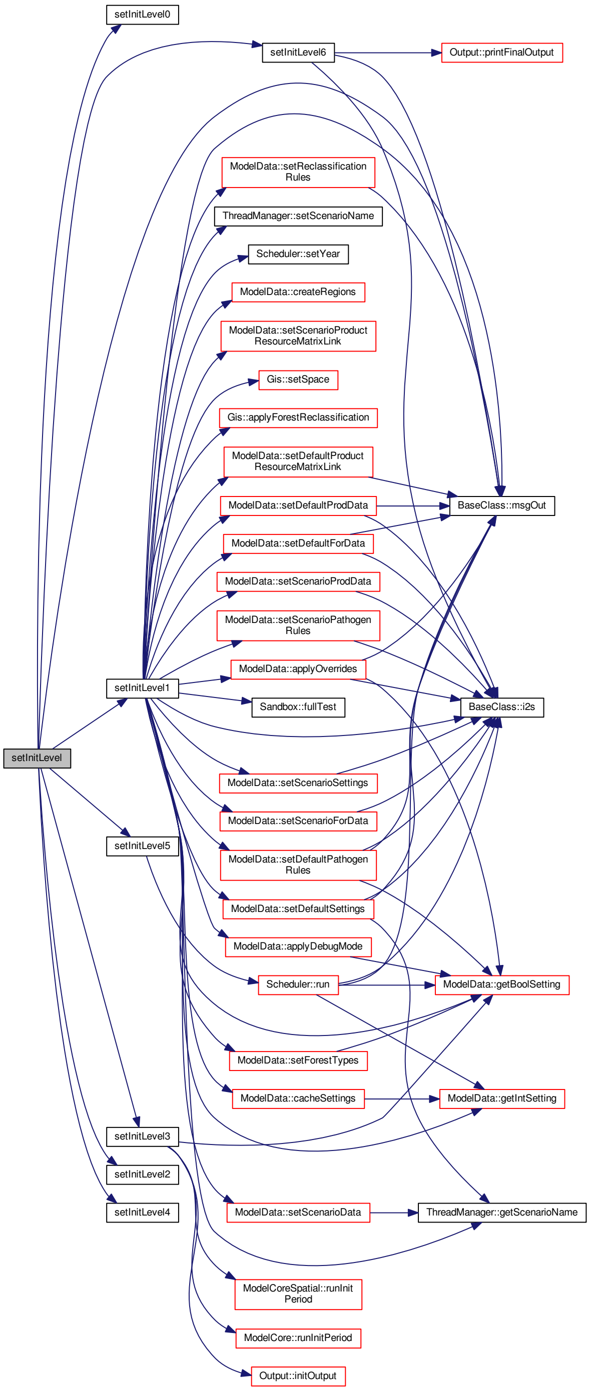
Wrapper to the correct setInitLevelX()

Definition at line 47 of file Init.cpp.

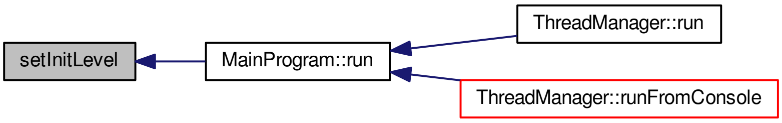
Referenced by MainProgram::run().

```
00047                               {
00048 
00049   switch (level_h){
00050     case 0:
00051       this->setInitLevel0();
00052       break;
00053     case 1:
00054       this->setInitLevel1();
00055       break;
00056     case 2:
00057       this->setInitLevel2();
00058       break;
00059     case 3:
00060       this->setInitLevel3();
00061       break;
00062     case 4:
00063       this->setInitLevel4();
00064       break;
00065     case 5:
00066       this->setInitLevel5();
00067       break;
00068     case 6:
00069       this->setInitLevel6();
00070       break;
00071     default:
00072       msgOut(MSG_ERROR,"unexpected Init level");
00073     }
00074 }
```

Here is the call graph for this function:


setInitLevel0
setInitLevel6
Output::printFinalOutput
ModelData::setReclassification Rules
ThreadManager::setScenarioName
Scheduler::setYear
ModelData::createRegions
ModelData::setScenarioProduct ResourceMatrixLink
Gis::setSpace
Gis::applyForestReclassification
ModelData::setDefaultProduct ResourceMatrixLink
ModelData::setDefaultProdData
BaseClass::msgOut
ModelData::setDefaultForData
ModelData::setScenarioProdData
ModelData::setScenarioPathogen Rules
ModelData::applyOverrides
setInitLevel1
Sandbox::fullTest
BaseClass::i2s
setInitLevel
ModelData::setScenarioSettings
ModelData::setScenarioForData
setInitLevel5
ModelData::setDefaultPathogen Rules
ModelData::setDefaultSettings
ModelData::applyDebugMode
Scheduler::run
ModelData::getBoolSetting
ModelData::setForestTypes
ModelData::cacheSettings
ModelData::getIntSetting
setInitLevel3
setInitLevel2
setInitLevel4
ModelData::setScenarioData
ThreadManager::getScenarioName
ModelCoreSpatial::runInit Period
ModelCore::runInitPeriod
Output::initOutput

Here is the caller graph for this function:



4.14.3.3 void setInitLevel0 ( )

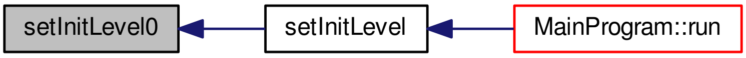
Unused, reserver for future use.

Definition at line 77 of file Init.cpp.

Referenced by setInitLevel().

```
00077                      {
00078   //unused now
00079   InitState=0;
00080 }
```

Here is the caller graph for this function:



4.14.3.4 void setInitLevel1 ( )

Setting up the space, the model objects and the agents (definitions only)

Setting up the space
Level 1 :

- set the environment (settings, available resource name, possible activities)
- init the space

See also

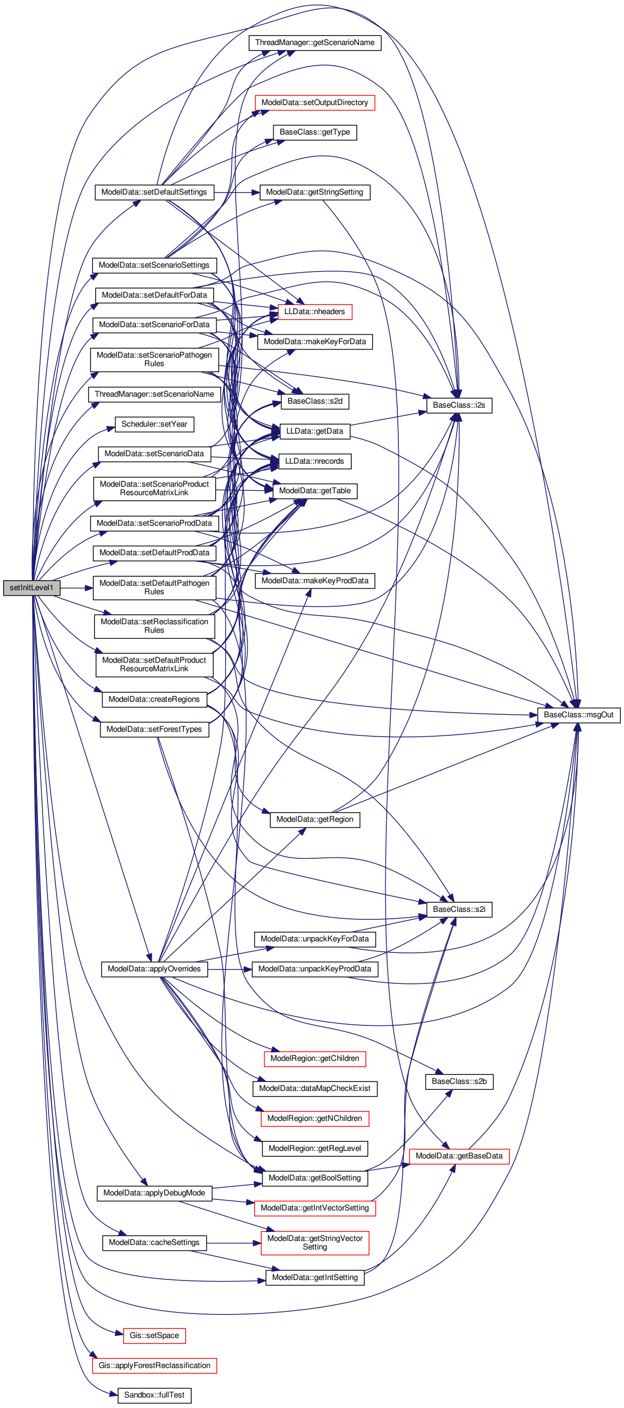
ModelData::setDefaultSettings();
Gis::setSpace()
Manager_farmers::setAgentMoulds()

Definition at line 93 of file Init.cpp.

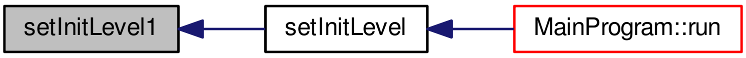
Referenced by setInitLevel().

```
00093                   {
00094   //Loading data from file.
00095   InitState=1;
00096   msgOut(MSG_DEBUG,"Entering Init state "+i2s(InitState));
00097   time(&now);
00098   current = localtime(&now);
00099   string timemessage = "Local time is "+i2s(current->tm_hour)+":"+i2s(
      current->tm_min)+":"+ i2s(current->tm_sec);
00100   msgOut(MSG_INFO, timemessage);
00101   string scenarioName = MTHREAD->getScenarioName();
00102   MTHREAD->MD->setScenarioData(); // set the characteristics (including overriding
       tables of the scneario)
00103   MTHREAD->MD->setDefaultSettings();
00104   MTHREAD->MD->setScenarioSettings();
00105   if(MTHREAD->MD->getBoolSetting("newRandomSeed")){
00106     // See here for how to use the new C++11 random functions:
00107     // http://www.johndcook.com/cpp_TR1_random.html
00108     // usage example:
00109     // std::normal_distribution<double> d(100000,3);
00110     // double x = d(*MTHREAD->gen);
00111     srand(time(NULL));
00112     //std::random_device randev;
00113     //MTHREAD->gen = new std::mt19937(randev());
00114     MTHREAD->gen = new std::mt19937(time(0));
00115 
00116     //TO.DO change scenarioname to scenarioname_random number
00117     uniform_int_distribution<> ud(1, 1000000);
00118     int randomscenario = ud(*MTHREAD->gen);
00119 
00120     MTHREAD->setScenarioName(scenarioName+"_"+i2s(randomscenario));
00121 
00122   } else {
00123     MTHREAD->gen = new std::mt19937(NULL);
00124   }
00125   MTHREAD->SCD->setYear(MTHREAD->MD->getIntSetting("initialYear"));
00126   MTHREAD->MD->cacheSettings();
00127 
00128   MTHREAD->MD->createRegions();
00129   MTHREAD->MD->setDefaultForData();
00130   MTHREAD->MD->setScenarioForData();
00131   MTHREAD->MD->setDefaultProdData();
00132   MTHREAD->MD->setScenarioProdData();
00133   MTHREAD->MD->setForestTypes();
00134   MTHREAD->MD->setReclassificationRules();
00135   MTHREAD->MD->applyOverrides(); // Cancel all reg1 level data and trasform them in
       reg2 level if not already existing. Acts on forDataMap, prodDataMap and reclRules vectors
00136   MTHREAD->MD->setDefaultPathogenRules();
00137   MTHREAD->MD->setScenarioPathogenRules();
00138   MTHREAD->MD->setDefaultProductResourceMatrixLink();
00139   MTHREAD->MD->setScenarioProductResourceMatrixLink();
00140   MTHREAD->MD->applyDebugMode();
00141   MTHREAD->GIS->setSpace();
00142   MTHREAD->GIS->applyForestReclassification();
00143   MTHREAD->TEST->fullTest(); // normally empty function
00144 }
```

Here is the call graph for this function:


setInitLevel1
ThreadManager::getScenarioName
ModelData::setOutputDirectory
BaseClass::getType
ModelData::setDefaultSettings
ModelData::getStringSetting
ModelData::setScenarioSettings
ModelData::setDefaultForData
LLData::nheaders
ModelData::setScenarioForData
ModelData::makeKeyForData
ModelData::setScenarioPathogenRules
ThreadManager::setScenarioName
BaseClass::s2d
BaseClass::i2s
Scheduler::setYear
LLData::getData
ModelData::setScenarioData
LLData::nrecords
ModelData::setScenarioProductResourceMatrixLink
ModelData::getTable
ModelData::setScenarioProdData
ModelData::setDefaultProdData
ModelData::makeKeyProdData
ModelData::setDefaultPathogenRules
ModelData::setReclassificationRules
ModelData::setDefaultProductResourceMatrixLink
BaseClass::msgOut
ModelData::createRegions
ModelData::setForestTypes
ModelData::getRegion
BaseClass::s2i
ModelData::unpackKeyForData
ModelData::applyOverrides
ModelData::unpackKeyProdData
ModelRegion::getChildren
BaseClass::s2b
ModelData::dataMapCheckExist
ModelRegion::getNChildren
ModelRegion::getRegLevel
ModelData::getBaseData
ModelData::getBoolSetting
ModelData::applyDebugMode
ModelData::getIntVectorSetting
ModelData::getStringVectorSetting
ModelData::cacheSettings
ModelData::getIntSetting
Gis::setSpace
Gis::applyForestReclassification
Sandbox::fullTest

Here is the caller graph for this function:



4.14.3.5 void setInitLevel2 ( )

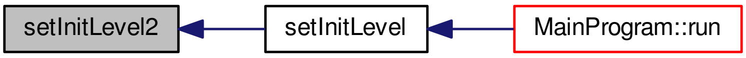
Unused, reserver for future use.

Definition at line 147 of file Init.cpp.

Referenced by setInitLevel().

```
00147                      {
00148   InitState=2;
00149 }
```

Here is the caller graph for this function:



4.14.3.6 void setInitLevel3 ( )

Linking object to agents and assigning space proprieties to objects and agents.

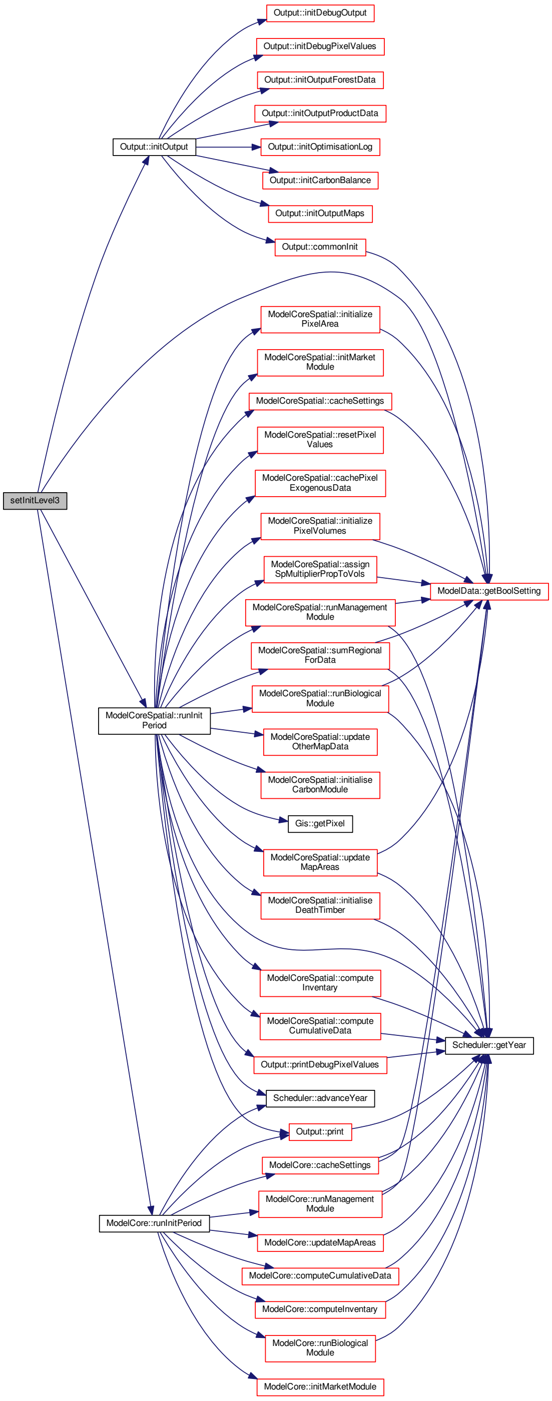
Init 3 run the simulation/assign the values for the pre-optimisation year(s)

Definition at line 155 of file Init.cpp.

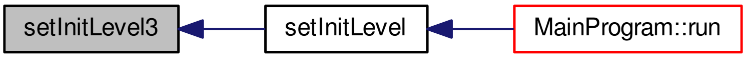
Referenced by setInitLevel().

```
00155                      {
00156   InitState=3;
00157   MTHREAD->DO->initOutput();        // initialize the output files
00158   if(MTHREAD->MD->getBoolSetting("usePixelData")){
00159     MTHREAD->SCORE->runInitPeriod();
00160   } else {
00161     MTHREAD->CORE->runInitPeriod();
00162   }
00163 }
```

Here is the call graph for this function:


setInitLevel3
Output::initOutput
Output::initDebugOutput
Output::initDebugPixelValues
Output::initOutputForestData
Output::initOutputProductData
Output::initOptimisationLog
Output::initCarbonBalance
Output::initOutputMaps
Output::commonInit
ModelCoreSpatial::initialize PixelArea
ModelCoreSpatial::initMarket Module
ModelCoreSpatial::cacheSettings
ModelCoreSpatial::resetPixel Values
ModelCoreSpatial::cachePixel ExogenousData
ModelCoreSpatial::initialize PixelVolumes
ModelCoreSpatial::assign SpMultiplierPropToVols
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::sumRegional ForData
ModelCoreSpatial::runBiological Module
ModelCoreSpatial::update OtherMapData
ModelCoreSpatial::initialise CarbonModule
Gis::getPixel
ModelCoreSpatial::update MapAreas
ModelCoreSpatial::initialise DeathTimber
ModelCoreSpatial::compute Inventary
ModelCoreSpatial::compute CumulativeData
Output::printDebugPixelValues
Scheduler::advanceYear
Output::print
ModelCore::cacheSettings
ModelCore::runManagement Module
ModelCore::updateMapAreas
ModelCore::computeCumulativeData
ModelCore::computeInventary
ModelCore::runBiological Module
ModelCore::initMarketModule
ModelCoreSpatial::runInit Period
ModelCore::runInitPeriod
ModelData::getBoolSetting
Scheduler::getYear

Here is the caller graph for this function:



4.14.3.7 void setInitLevel4 ( )

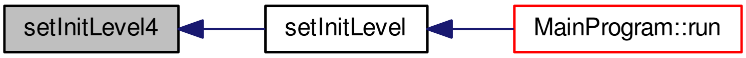
Unused, reserver for future use.

Definition at line 166 of file Init.cpp.

Referenced by setInitLevel().

```
00166                     {
00167   InitState=4;
00168 }
```

Here is the caller graph for this function:



4.14.3.8 void setInitLevel5 ( )

Simulation start.

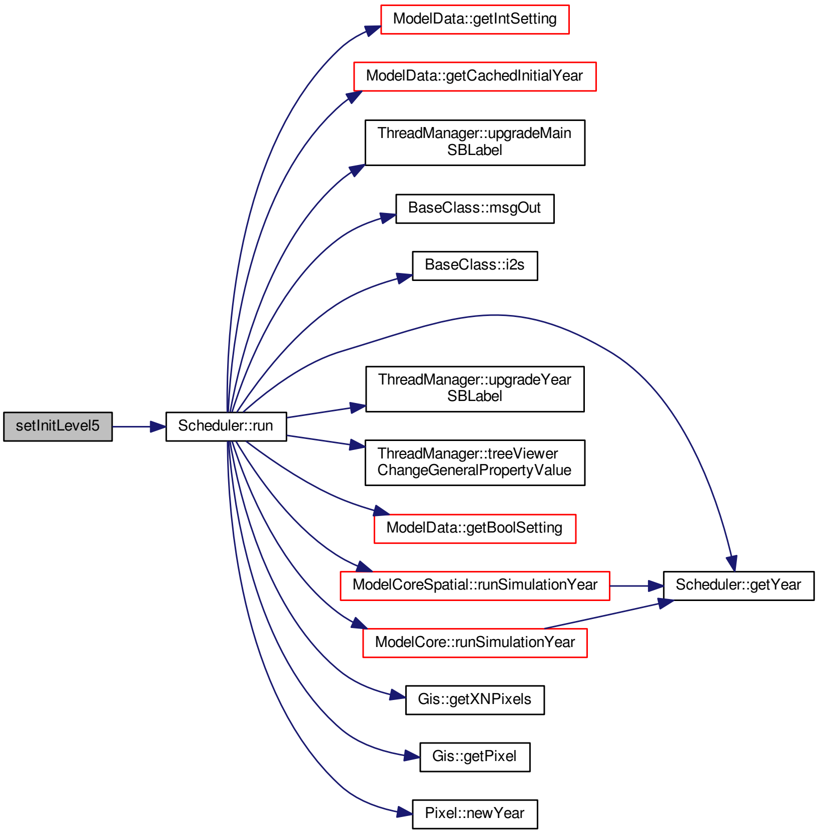
Init level 5 pass the controll to the Scheduler object for the running of the simulations.

Definition at line 174 of file Init.cpp.

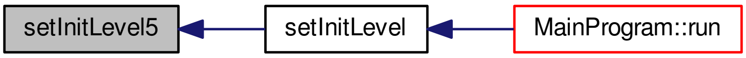
Referenced by setInitLevel().

```
00174                     {
00175   InitState=5;
00176   MTHREAD->SCD->run(); // !!!! go "bello" !!!! start the simulation !!!!!
00177 }
```

Here is the call graph for this function:


ModelData::getIntSetting
ModelData::getCachedInitialYear
ThreadManager::upgradeMain SBLabel
BaseClass::msgOut
BaseClass::i2s
ThreadManager::upgradeYear SBLabel
setInitLevel5
Scheduler::run
ThreadManager::treeViewer ChangeGeneralPropertyValue
ModelData::getBoolSetting
ModelCoreSpatial::runSimulationYear
Scheduler::getYear
ModelCore::runSimulationYear
Gis::getXNPixels
Gis::getPixel
Pixel::newYear

Here is the caller graph for this function:



4.14.3.9 void setInitLevel6 ( )

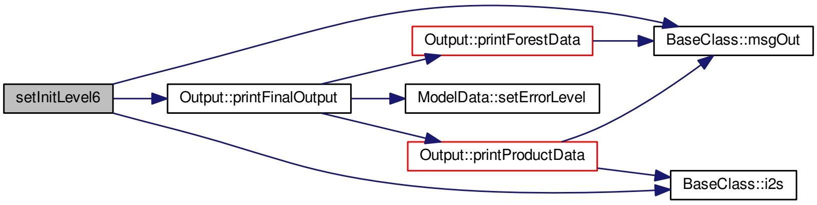
End of simulation (e.g. print summary statistics)

Definition at line 180 of file Init.cpp.

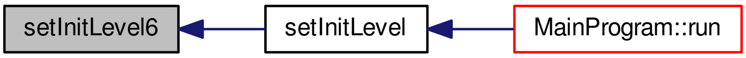
Referenced by setInitLevel().

```
                {
  InitState=6;
  MTHREAD->DO->printFinalOutput();
  msgOut(MSG_INFO, "Model has ended scheduled simulation in a regular way.");
  time(&now);
  current = localtime(&now);
  string timemessage = "Local time is "+i2s(current->tm_hour)+":"+i2s(
      current->tm_min)+":"+ i2s(current->tm_sec);
  msgOut(MSG_INFO, timemessage);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.14.4 Member Data Documentation

4.14.4.1 struct tm∗ current [private]

Definition at line 71 of file Init.h.

Referenced by setInitLevel1(), and setInitLevel6().

4.14.4.2 int InitState [private]

One of the 7 possible init states (0..6)

Definition at line 67 of file Init.h.

Referenced by Init(), setInitLevel0(), setInitLevel1(), setInitLevel2(), setInitLevel3(), setInitLevel4(), setInitLevel5(), and setInitLevel6().

4.14.4.3 time_t now [private]

Definition at line 72 of file Init.h.

Referenced by setInitLevel1(), and setInitLevel6().

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

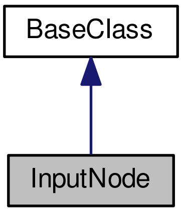
- /home/lobianco/git/ffsm_pp/src/Init.h
- /home/lobianco/git/ffsm_pp/src/Init.cpp

4.15 InputNode Class Reference

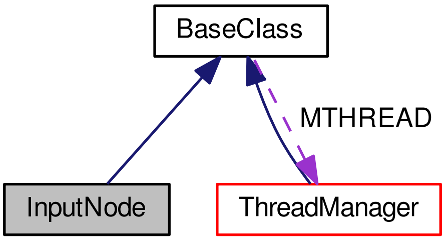
Wrapper around the underlying library for reading DOM elements (nodes).

`#include <InputNode.h>`

Inheritance diagram for InputNode:



Collaboration diagram for InputNode:



Public Member Functions

- InputNode ()
- InputNode (QDomElement domElement_h)
- ∼InputNode ()
- bool setWorkingFile (std::string filename_h)

  Load the file on memory. Return false if no success.
- int getIntContent ()

  Get the content between its tagName as integer.
- double getDoubleContent ()

  Get the content between its tagName as double.
- string getStringContent ()

  Get the content between its tagName as std::string.
- bool getBoolContent ()

  Get the content between its tagName as bool.
- int getIntAttributeByName (string attributeName_h)

  Get an attribute by name as integer.
- double getDoubleAttributeByName (string attributeName_h)

  Get an attribute by name as double.
- string getStringAttributeByName (string attributeName_h)

  Get an attribute by name as string.
- bool hasAttributeByName (string attributeName_h)

  Check if an attribute with a certain name exist.
- InputNode getNodeByName (string nodeName_h, int debugLevel=MSG_CRITICAL_ERROR, bool child←↩ Flag=false)

  return 0-or-1 nodes by name.
- vector< InputNode > getNodesByName (string nodeName_h, int debugLevel=MSG_WARNING, bool child←↩ Flag=false)
- vector< InputNode > getChildNodes ()

  Retrieve a child node with gived name and optionally with gived attribute or gived pair attribute/value. It raises an error if more than one.
- bool hasChildNode (string name_h)

  True if it has specified child node.
- int getChildNodesCount ()

  *Only **Elements***
- string getNodeName ()

Private Attributes

- QDomElement domElement

  The underlying library-depending DOM rappresentation of the element.

Additional Inherited Members

4.15.1 Detailed Description

Wrapper around the underlying library for reading DOM elements (nodes).

A small wrapper class using an underlying library (currently QtXml) to read DOM nodes.
This class works with the individual nodes (DOM Elements), while the companion class InputDocument wrapper the whole document (DOM Document).
Note: In the DOM terminology "Elements" are a subset of the more general "nodes" (that include comments and other typologies..)

Author

Antonello Lobianco

Definition at line 51 of file InputNode.h.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 InputNode ( )

Definition at line 30 of file InputNode.cpp.

```
00030                        {
00031 }
```

4.15.2.2 InputNode (QDomElement *domElement_h*) `[inline]`

Definition at line 55 of file InputNode.h.

```
00055 {domElement=domElement_h;}; //<Constructor
```

4.15.2.3 ∼InputNode ( )

Definition at line 33 of file InputNode.cpp.

```
00033                         {
00034 }
```

4.15.3 Member Function Documentation

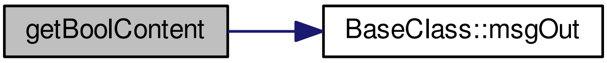
4.15.3.1 bool getBoolContent ( )

Get the content between its tagName as bool.

Definition at line 79 of file InputNode.cpp.

```
00079                            {
00080   string content = domElement.text().toStdString();
00081   if (content == "false" || content == "falso" || content == "FALSE" || content == "0")
00082     return false;
00083   else if (content == "true" || content == "vero" || content == "TRUE" || content == "1")
00084     return true;
00085   msgOut(MSG_WARNING, "Sorry, I don't know how to convert "+content+" to a bool value. I
      return true... hope for the best");
00086   return true;
00087 }
```

Here is the call graph for this function:



4.15.3.2 vector< InputNode > getChildNodes ( )

Retrieve a child node with gived name and optionally with gived attribute or gived pair attribute/value. It raises an error if more than one.

Retrieve all child nodes with gived name and optionally with gived attribute or gived pair attribute/value. It raises an error if more than one. Filtered to return only child **Elements**

Definition at line 235 of file InputNode.cpp.

```
00235                          {
00236   vector <InputNode> myNodeVector;
00237   QDomNodeList myElementList = domElement.childNodes();
00238   for (int i=0;i<myElementList.size();i++){
00239     if (myElementList.item(i).nodeType() == QDomNode::ElementNode ){
00240       InputNode myInputNode(myElementList.item(i).toElement());
00241       myNodeVector.push_back(myInputNode);
00242     }
00243   }
00244   return myNodeVector;
00245 }
```

4.15.3.3 int getChildNodesCount ( )

Only **Elements**

Definition at line 260 of file InputNode.cpp.

```
00260                               {
00261   int myElementListCountInt = 0;
00262   QDomNodeList myElementList = domElement.childNodes();
00263   for (int i=0;i<myElementList.size();i++){
00264     if (myElementList.item(i).nodeType() == QDomNode::ElementNode ){
00265       myElementListCountInt++ ;
00266     }
00267   }
00268   return myElementListCountInt;
00269 }
```

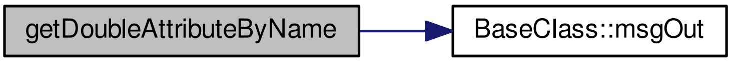
4.15.3.4 double getDoubleAttributeByName (std::string *attributeName_h*)

Get an attribute by name as double.

Definition at line 100 of file InputNode.cpp.

```
00100                                                               {
00101   if (domElement.hasAttribute(attributeName_h.c_str())){
00102     return domElement.attribute(attributeName_h.c_str()).toDouble();
00103   }else{
00104     msgOut(MSG_ERROR, "Element doens't have attribute " + attributeName_h );
00105     return 0;
00106   }
00107 }
```

Here is the call graph for this function:



4.15.3.5 double getDoubleContent ( )

Get the content between its tagName as double.

Definition at line 69 of file InputNode.cpp.

```
00069                             {
00070   return domElement.text().toDouble(); // This is a Qt function that works both with dot and
       comma separators !
00071 }
```

4.15.3.6 int getIntAttributeByName (std::string *attributeName_h*)

Get an attribute by name as integer.

Definition at line 90 of file InputNode.cpp.

```
00090                                                          {
00091   if (domElement.hasAttribute(attributeName_h.c_str())){
00092     return domElement.attribute(attributeName_h.c_str()).toInt();
00093   }else{
00094     msgOut(MSG_ERROR, "Element doens't have attribute " + attributeName_h );
00095     return 0;
00096   }
00097 }
```

Here is the call graph for this function:

getIntAttributeByName
BaseClass::msgOut

4.15.3.7 int getIntContent ( )

Get the content between its tagName as integer.

Definition at line 64 of file InputNode.cpp.

```
00064                          {
00065   return domElement.text().toInt();
00066 }
```

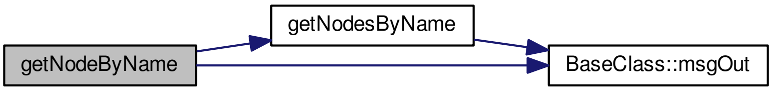
4.15.3.8 InputNode getNodeByName (string *nodeName_h,* int *debugLevel* = MSG_CRITICAL_ERROR*,* bool *childFlag* = `false`)

return 0-or-1 nodes by name.

Definition at line 129 of file InputNode.cpp.

```
00129                                                                         {
00130   /*
00131   QDomNodeList myElementList = domElement.elementsByTagName ( nodeName_h.c_str() );
00132   if (myElementList.size()>1){
00133     msgOut(debugLevel, "Too many elements. Expected only one of type "+nodeName_h);
00134   }
00135   if (myElementList.isEmpty()){
00136     msgOut(debugLevel, "No elements in the XML file. Expected 1 of type "+nodeName_h);
00137   }
00138   QDomElement myElement = myElementList.item(0).toElement() ;
00139   InputNode myInputNode(myElement);
00140   return myInputNode; */
00141   vector<InputNode> myNodes = getNodesByName(nodeName_h, debugLevel, childFlag);
00142   if (myNodes.size()>1){
00143     msgOut(debugLevel, "Too many elements. Expected only one of type "+nodeName_h);
00144     return myNodes[0];
00145   }
00146   if (myNodes.size() == 0){
00147     msgOut(debugLevel, "No elements in the XML file. Expected 1 of type "+nodeName_h+". Returning
       emty node!!");
00148     InputNode toReturn;
00149     return toReturn;
00150   }
00151   return myNodes[0];
00152 }
```

Here is the call graph for this function:



4.15.3.9 string getNodeName ( )

Definition at line 272 of file InputNode.cpp.

```
00272                         {
00273   return domElement.tagName().toStdString();
00274 }
```

4.15.3.10 vector< InputNode > getNodesByName (string *nodeName_h,* int *debugLevel* = MSG_WARNING*,* bool *childFlag* = `false`)

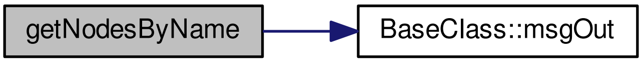
return 0-to-n nodes by name

Definition at line 155 of file InputNode.cpp.

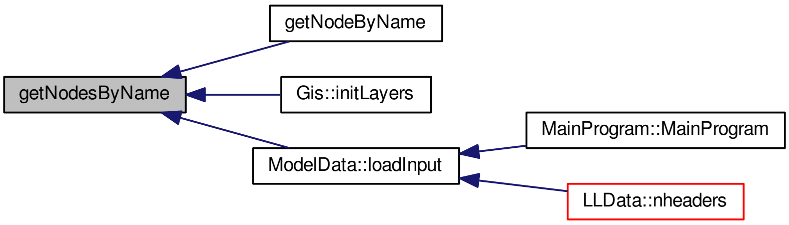
Referenced by getNodeByName(), Gis::initLayers(), and ModelData::loadInput().

```
                                     {
 vector <InputNode> myNodeVector;
 if (!childFlag){
  QDomNodeList myElementList = domElement.elementsByTagName ( nodeName_h.c_str() );
  for (int i=0;i<myElementList.size();i++){
   InputNode myInputNode(myElementList.item(i).toElement());
   myNodeVector.push_back(myInputNode);
  }

 }
 else {
  QDomNodeList myElementList = domElement.childNodes();
  for (int i=0;i<myElementList.size();i++){
   if (   myElementList.item(i).nodeType() == QDomNode::ElementNode
     && myElementList.item(i).toElement().tagName().toStdString() == nodeName_h){
    InputNode myInputNode(myElementList.item(i).toElement());
    myNodeVector.push_back(myInputNode);
   }
  }
 }
 if (myNodeVector.size()==0){
  msgOut(debugLevel, "No elements in the XML file. Expected at least one of type "+nodeName_h);
 }
 //for (int i=0;i<myElementList.size();i++){
 //  InputNode myInputNode(myElementList.item(i).toElement());
 //  myNodeVector.push_back(myInputNode);

  /*InputNode myInputNode(myElementList.item(i).toElement());
  string firstNodeContent= myInputNode.getStringContent();
  // checking that the setting is not an empy line nor a comment (e.g. starting with "#")..
  if(firstNodeContent=="") continue;
  unsigned int z;
  z = firstNodeContent.find("#");
  if( z!=string::npos && z == 0) continue;
  // chacking also the "childs" as in the XMLs deriving from csv I want delete the whole "<record>" tree, including his childs (fields)
  vector <InputNode> childs = myInputNode.getChildNodes();
  if(childs.size()>0){
   string firstChildContent= childs[0].getStringContent();
   // checking that the setting is not an empy line nor a comment (e.g. starting with "#")..
   if(firstChildContent=="") continue;
   unsigned int y;
   y = firstChildContent.find("#");
   if( y!=string::npos && y == 0) continue;
  }
  myNodeVector.push_back(myInputNode);
  */


 //}
 return myNodeVector;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



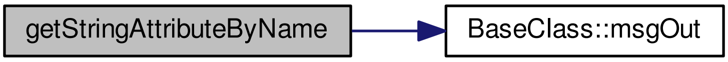
4.15.3.11 string getStringAttributeByName (std::string *attributeName_h*)

Get an attribute by name as string.

Definition at line 110 of file InputNode.cpp.

```
00110                                                           {
00111   if (domElement.hasAttribute(attributeName_h.c_str())){
00112     return domElement.attribute(attributeName_h.c_str()).toStdString();
00113   }else{
00114     msgOut(MSG_ERROR, "Element doens't have attribute " + attributeName_h );
00115     return "";
00116   }
00117 }
```

Here is the call graph for this function:



4.15.3.12 std::string getStringContent ( )

Get the content between its tagName as std::string.

Definition at line 74 of file InputNode.cpp.

```
00074                             {
00075   return domElement.text().toStdString();
00076 }
```

4.15.3.13 bool hasAttributeByName (std::string *attributeName_h*)

Check if an attribute with a certain name exist.

Definition at line 120 of file InputNode.cpp.

```
00120                                                          {
00121   if (domElement.hasAttribute(attributeName_h.c_str())){
00122     return 1;
00123   }else{
00124     return 0;
00125   }
00126 }
```

4.15.3.14 bool hasChildNode (string *name_h*)

True if it has specified child node.

Definition at line 248 of file InputNode.cpp.

```
00248                                    {
00249   bool toReturn = false;
00250   QDomNodeList myElementList = domElement.childNodes();
00251   for (int i=0;i<myElementList.size();i++){
00252     if (myElementList.item(i).nodeType() == QDomNode::ElementNode ){
00253       if(myElementList.item(i).toElement().tagName().toStdString() == name_h) return true;
00254     }
00255   }
00256   return toReturn;
00257 }
```

4.15.3.15 bool setWorkingFile (std::string *filename_h*)

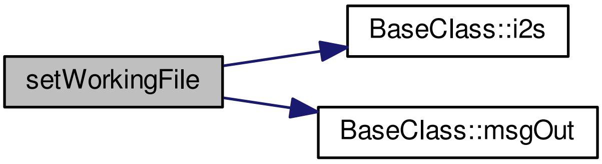
Load the file on memory. Return false if no success.

Definition at line 37 of file InputNode.cpp.

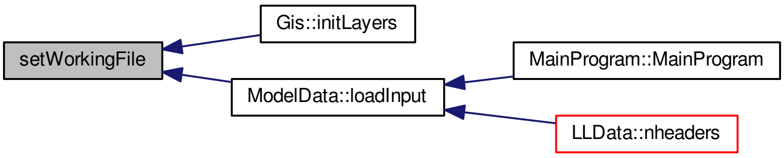
Referenced by Gis::initLayers(), and ModelData::loadInput().

```
00037                                             {
00038
00039   QString errorStr;
00040   int errorLine;
00041   int errorColumn;
00042
00043   QFile file(filename_h.c_str());
00044   QIODevice* device;
00045   device = &file;
00046
00047   QDomDocument doc;
00048   if (!doc.setContent(device, true, &errorStr, &errorLine, &errorColumn)) {
00049     string message = "XML error on file "+ filename_h + " at line ";
00050     message.append(i2s(errorLine));
00051     message.append(" column ");
00052     message = message.c_str() + i2s(errorColumn);
00053     message = message + ": ";
00054     message = message + errorStr.toStdString();
00055     msgOut(MSG_WARNING, message.c_str());
00056     return false;
00057     }
00058   domElement = doc.documentElement();
00059   return true;
00060 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.15.4 Member Data Documentation

4.15.4.1 QDomElement domElement [private]

The underlying library-depending DOM rappresentation of the element.

Definition at line 80 of file InputNode.h.

Referenced by getBoolContent(), getChildNodes(), getChildNodesCount(), getDoubleAttributeByName(), getDoubleContent(), getIntAttributeByName(), getIntContent(), getNodeName(), getNodesByName(), getStringAttributeByName(), getStringContent(), hasAttributeByName(), hasChildNode(), and setWorkingFile().

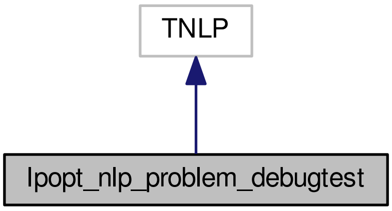
The documentation for this class was generated from the following files:

- /home/lobianco/git/ffsm_pp/src/InputNode.h
- /home/lobianco/git/ffsm_pp/src/InputNode.cpp

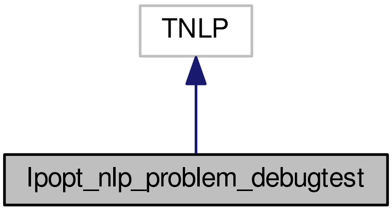
4.16 Ipopt_nlp_problem_debugtest Class Reference

`#include <Ipopt_nlp_problem_debugtest.h>`

Inheritance diagram for Ipopt_nlp_problem_debugtest:



Collaboration diagram for Ipopt_nlp_problem_debugtest:



Public Member Functions

- Ipopt_nlp_problem_debugtest ()
- virtual ∼Ipopt_nlp_problem_debugtest ()

Overloaded from TNLP

- virtual bool get_nlp_info (Index &n, Index &m, Index &nnz_jac_g, Index &nnz_h_lag, IndexStyleEnum &index_style)
- virtual bool get_bounds_info (Index n, Number ∗x_l, Number ∗x_u, Index m, Number ∗g_l, Number ∗g_u)
- virtual bool get_starting_point (Index n, bool init_x, Number ∗x, bool init_z, Number ∗z_L, Number ∗z_U, Index m, bool init_lambda, Number ∗lambda)
- virtual bool eval_f (Index n, const Number ∗x, bool new_x, Number &obj_value)
- virtual bool eval_grad_f (Index n, const Number ∗x, bool new_x, Number ∗grad_f)
- virtual bool eval_g (Index n, const Number ∗x, bool new_x, Index m, Number ∗g)
- virtual bool eval_jac_g (Index n, const Number ∗x, bool new_x, Index m, Index nele_jac, Index ∗iRow, Index ∗jCol, Number ∗values)

- virtual bool eval_h (Index n, const Number ∗x, bool new_x, Number obj_factor, Index m, const Number ∗lambda, bool new_lambda, Index nele_hess, Index ∗iRow, Index ∗jCol, Number ∗values)

Solution Methods

- virtual void finalize_solution (SolverReturn status, Index n, const Number ∗x, const Number ∗z_L, const Number ∗z_U, Index m, const Number ∗g, const Number ∗lambda, Number obj_value, const IpoptData ∗ip_data, IpoptCalculatedQuantities ∗ip_cq)

Private Member Functions

Methods to block default compiler methods.

The compiler automatically generates the following three methods. Since the default compiler implementation is generally not what you want (for all but the most simple classes), we usually put the declarations of these methods in the private section and never implement them. This prevents the compiler from implementing an incorrect "default" behavior without us knowing. (See Scott Meyers book, "Effective C++")

- Ipopt_nlp_problem_debugtest (const Ipopt_nlp_problem_debugtest &)
- Ipopt_nlp_problem_debugtest & operator= (const Ipopt_nlp_problem_debugtest &)

4.16.1 Detailed Description

C++ Example NLP for interfacing a problem with IPOPT. HS071_NLP implements a C++ example of problem 71 of the Hock-Schittkowski test suite. This example is designed to go along with the tutorial document and show how to interface with IPOPT through the TNLP interface.

Problem hs071 looks like this

```
min   x1*x4*(x1 + x2 + x3)  +  x3
s.t.  x1*x2*x3*x4                  >=  25
      x1**2 + x2**2 + x3**2 + x4**2  =  40
      1 <=  x1,x2,x3,x4  <= 5

Starting point:
   x = (1, 5, 5, 1)

Optimal solution:
   x = (1.00000000, 4.74299963, 3.82114998, 1.37940829)
```

Definition at line 29 of file Ipopt_nlp_problem_debugtest.h.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 Ipopt_nlp_problem_debugtest ( )

default constructor

Definition at line 9 of file Ipopt_nlp_problem_debugtest.cpp.

```
00010 {}
```

4.16.2.2 ∼Ipopt_nlp_problem_debugtest ( ) [virtual]

default destructor

Definition at line 13 of file Ipopt_nlp_problem_debugtest.cpp.

```
{}
```

4.16.2.3 Ipopt_nlp_problem_debugtest (const Ipopt_nlp_problem_debugtest &) [private]

4.16.3 Member Function Documentation

4.16.3.1 bool eval_f (Index *n*, const Number ∗ *x*, bool *new_x*, Number & *obj_value*) [virtual]

Method to return the objective value

Definition at line 96 of file Ipopt_nlp_problem_debugtest.cpp.

```
{
  assert(n == 4);

  obj_value = x[0] * x[3] * (x[0] + x[1] + x[2]) + x[2];

  return true;
}
```

4.16.3.2 bool eval_g (Index *n*, const Number ∗ *x*, bool *new_x*, Index *m*, Number ∗ *g*) [virtual]

Method to return the constraint residuals

Definition at line 119 of file Ipopt_nlp_problem_debugtest.cpp.

```
{
  assert(n == 4);
  assert(m == 2);

  g[0] = x[0] * x[1] * x[2] * x[3];
  g[1] = x[0]*x[0] + x[1]*x[1] + x[2]*x[2] + x[3]*x[3];

  return true;
}
```

4.16.3.3 bool eval_grad_f (Index *n*, const Number ∗ *x*, bool *new_x*, Number ∗ *grad_f*) [virtual]

Method to return the gradient of the objective

Definition at line 106 of file Ipopt_nlp_problem_debugtest.cpp.

```
{
  assert(n == 4);

  grad_f[0] = x[0] * x[3] + x[3] * (x[0] + x[1] + x[2]);
  grad_f[1] = x[0] * x[3];
  grad_f[2] = x[0] * x[3] + 1;
  grad_f[3] = x[0] * (x[0] + x[1] + x[2]);

  return true;
}
```

4.16.3.4 bool eval_h (Index *n*, const Number ∗ *x*, bool *new_x*, Number *obj_factor*, Index *m*, const Number ∗ *lambda*, bool *new_lambda*, Index *nele_hess*, Index ∗ *iRow*, Index ∗ *jCol*, Number ∗ *values*) [virtual]

Method to return: 1) The structure of the hessian of the lagrangian (if "values" is NULL) 2) The values of the hessian of the lagrangian (if "values" is not NULL)

Definition at line 175 of file Ipopt_nlp_problem_debugtest.cpp.

```
{
  if (values == NULL) {
    // return the structure. This is a symmetric matrix, fill the lower left
    // triangle only.

    // the hessian for this problem is actually dense
    Index idx=0;
    for (Index row = 0; row < 4; row++) {
      for (Index col = 0; col <= row; col++) {
        iRow[idx] = row;
        jCol[idx] = col;
        idx++;
      }
    }

    assert(idx == nele_hess);
  }
  else {
    // return the values. This is a symmetric matrix, fill the lower left
    // triangle only

    // fill the objective portion
    values[0] = obj_factor * (2*x[3]); // 0,0

    values[1] = obj_factor * (x[3]);   // 1,0
    values[2] = 0.;                    // 1,1

    values[3] = obj_factor * (x[3]);   // 2,0
    values[4] = 0.;                    // 2,1
    values[5] = 0.;                    // 2,2

    values[6] = obj_factor * (2*x[0] + x[1] + x[2]); // 3,0
    values[7] = obj_factor * (x[0]);                 // 3,1
    values[8] = obj_factor * (x[0]);                 // 3,2
    values[9] = 0.;                                  // 3,3


    // add the portion for the first constraint
    values[1] += lambda[0] * (x[2] * x[3]); // 1,0

    values[3] += lambda[0] * (x[1] * x[3]); // 2,0
    values[4] += lambda[0] * (x[0] * x[3]); // 2,1

    values[6] += lambda[0] * (x[1] * x[2]); // 3,0
    values[7] += lambda[0] * (x[0] * x[2]); // 3,1
    values[8] += lambda[0] * (x[0] * x[1]); // 3,2

    // add the portion for the second constraint
    values[0] += lambda[1] * 2; // 0,0

    values[2] += lambda[1] * 2; // 1,1

    values[5] += lambda[1] * 2; // 2,2

    values[9] += lambda[1] * 2; // 3,3
  }

  return true;
}
```

4.16.3.5 bool eval_jac_g (Index *n*, const Number ∗ *x*, bool *new_x*, Index *m*, Index *nele_jac*, Index ∗ *iRow*, Index ∗ *jCol*, Number ∗ *values*) [virtual]

Method to return: 1) The structure of the jacobian (if "values" is NULL) 2) The values of the jacobian (if "values" is not NULL)

Definition at line 131 of file Ipopt_nlp_problem_debugtest.cpp.

```
{
  if (values == NULL) {
    // return the structure of the jacobian

    // this particular jacobian is dense
    iRow[0] = 0;
    jCol[0] = 0;
    iRow[1] = 0;
    jCol[1] = 1;
    iRow[2] = 0;
    jCol[2] = 2;
    iRow[3] = 0;
    jCol[3] = 3;
    iRow[4] = 1;
    jCol[4] = 0;
    iRow[5] = 1;
    jCol[5] = 1;
    iRow[6] = 1;
    jCol[6] = 2;
    iRow[7] = 1;
    jCol[7] = 3;
  }
  else {
    // return the values of the jacobian of the constraints

    values[0] = x[1]*x[2]*x[3]; // 0,0
    values[1] = x[0]*x[2]*x[3]; // 0,1
    values[2] = x[0]*x[1]*x[3]; // 0,2
    values[3] = x[0]*x[1]*x[2]; // 0,3

    values[4] = 2*x[0]; // 1,0
    values[5] = 2*x[1]; // 1,1
    values[6] = 2*x[2]; // 1,2
    values[7] = 2*x[3]; // 1,3
  }

  return true;
}
```

4.16.3.6 void finalize_solution (SolverReturn *status*, Index *n*, const Number ∗ *x*, const Number ∗ *z_L*, const Number ∗ *z_U*, Index *m*, const Number ∗ *g*, const Number ∗ *lambda*, Number *obj_value*, const IpoptData ∗ *ip_data*, IpoptCalculatedQuantities ∗ *ip_cq*) [virtual]

This method is called when the algorithm is complete so the TNLP can store/write the solution

Definition at line 241 of file Ipopt_nlp_problem_debugtest.cpp.

```
{
  // here is where we would store the solution to variables, or write to a file, etc
  // so we could use the solution.

  // For this example, we write the solution to the console
  std::cout << std::endl << std::endl << "Solution of the primal variables, x" << std::endl;
  for (Index i=0; i<n; i++) {
     std::cout << "x[" << i << "] = " << x[i] << std::endl;
  }

  std::cout << std::endl << std::endl << "Solution of the bound multipliers, z_L and z_U" << std::endl;
  for (Index i=0; i<n; i++) {
    std::cout << "z_L[" << i << "] = " << z_L[i] << std::endl;
  }
  for (Index i=0; i<n; i++) {
    std::cout << "z_U[" << i << "] = " << z_U[i] << std::endl;
  }

  std::cout << std::endl << std::endl << "Objective value" << std::endl;
  std::cout << "f(x*) = " << obj_value << std::endl;

  std::cout << std::endl << "Final value of the constraints:" << std::endl;
  for (Index i=0; i<m ;i++) {
    std::cout << "g(" << i << ") = " << g[i] << std::endl;
  }
}
```

4.16.3.7 bool get_bounds_info (Index *n*, Number ∗ *x_l*, Number ∗ *x_u*, Index *m*, Number ∗ *g_l*, Number ∗ *g_u*) [virtual]

Method to return the bounds for my problem

Definition at line 40 of file Ipopt_nlp_problem_debugtest.cpp.

```
{
  // here, the n and m we gave IPOPT in get_nlp_info are passed back to us.
  // If desired, we could assert to make sure they are what we think they are.
  assert(n == 4);
  assert(m == 2);

  // the variables have lower bounds of 1
  for (Index i=0; i<4; i++) {
    x_l[i] = 1.0;
  }

  // the variables have upper bounds of 5
  for (Index i=0; i<4; i++) {
    x_u[i] = 5.0;
  }

  // the first constraint g1 has a lower bound of 25
  g_l[0] = 25;
  // the first constraint g1 has NO upper bound, here we set it to 2e19.
  // Ipopt interprets any number greater than nlp_upper_bound_inf as
  // infinity. The default value of nlp_upper_bound_inf and nlp_lower_bound_inf
  // is 1e19 and can be changed through ipopt options.
  g_u[0] = 2e19;

  // the second constraint g2 is an equality constraint, so we set the
  // upper and lower bound to the same value
  g_l[1] = g_u[1] = 40.0;

  return true;
}
```

4.16.3.8 bool get_nlp_info (Index & *n*, Index & *m*, Index & *nnz_jac_g*, Index & *nnz_h_lag*, IndexStyleEnum & *index_style*) [virtual]

Method to return some info about the nlp

Definition at line 17 of file Ipopt_nlp_problem_debugtest.cpp.

```
{
  // The problem described in Ipopt_nlp_problem_debugtest.hpp has 4 variables, x[0] through x[3]
  n = 4;

  // one equality constraint and one inequality constraint
  m = 2;

  // in this example the jacobian is dense and contains 8 nonzeros
  nnz_jac_g = 8;

  // the hessian is also dense and has 16 total nonzeros, but we
  // only need the lower left corner (since it is symmetric)
  nnz_h_lag = 10;

  // use the C style indexing (0-based)
  index_style = TNLP::C_STYLE;

  return true;
}
```

4.16.3.9 bool get_starting_point (Index *n*, bool *init_x*, Number ∗ *x*, bool *init_z*, Number ∗ *z_L*, Number ∗ *z_U*, Index *m*, bool *init_lambda*, Number ∗ *lambda*) `[virtual]`

Method to return the starting point for the algorithm

Definition at line 74 of file Ipopt_nlp_problem_debugtest.cpp.

```
{
  // Here, we assume we only have starting values for x, if you code
  // your own NLP, you can provide starting values for the dual variables
  // if you wish
  assert(init_x == true);
  assert(init_z == false);
  assert(init_lambda == false);

  // initialize to the given starting point
  x[0] = 1.0;
  x[1] = 5.0;
  x[2] = 5.0;
  x[3] = 1.0;

  return true;
}
```

4.16.3.10 Ipopt_nlp_problem_debugtest& operator= (const Ipopt_nlp_problem_debugtest &) `[private]`

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

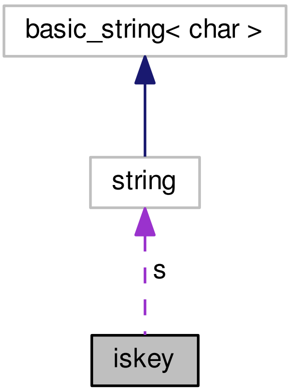
- /home/lobianco/git/ffsm_pp/src/Ipopt_nlp_problem_debugtest.h
- /home/lobianco/git/ffsm_pp/src/Ipopt_nlp_problem_debugtest.cpp

4.17 iskey Class Reference

Class to provide a simple integer-string key to be used in std maps.

`#include <BaseClass.h>`

Collaboration diagram for iskey:



Public Member Functions

- iskey ()
- iskey (int i_h, string s_h)
- ∼iskey ()
- bool operator== (const iskey &op2) const
- bool operator!= (const iskey &op2) const
- bool operator< (const iskey &op2) const
- bool operator> (const iskey &op2) const
- bool operator<= (const iskey &op2) const
- bool operator>= (const iskey &op2) const

Public Attributes

- int i
- string s

4.17.1 Detailed Description

Class to provide a simple integer-string key to be used in std maps.

Definition at line 176 of file BaseClass.h.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 iskey ( )

OTHER CLASSES THAN BASECLASS ////////////////// iskey class ///

Definition at line 403 of file BaseClass.cpp.

```
00403              {
00404   i = 0;
00405   s = "";
00406 }
```

4.17.2.2 iskey (int *i_h,* string *s_h*)

Definition at line 407 of file BaseClass.cpp.

```
00407                                  {
00408   i = i_h;
00409   s = s_h;
00410 }
```

4.17.2.3 ∼iskey ( )

Definition at line 412 of file BaseClass.cpp.

```
00412               {
00413
00414 }
```

4.17.3 Member Function Documentation

4.17.3.1 bool operator!= (const iskey & *op2*) const

Definition at line 425 of file BaseClass.cpp.

```
00425                                             {
00426   if(op2.i == i && op2.s == s){
00427     return false;
00428   }
00429   return true;
00430 }
```

4.17.3.2 bool operator< (const iskey & *op2*) const

Definition at line 433 of file BaseClass.cpp.

```
00433                                            {
00434   if (i < op2.i ) return true;
00435   if (i == op2.i) {
00436     if (s < op2.s) return true;
00437   }
00438   return false;
00439 }
```

4.17.3.3 bool operator<= (const iskey & *op2*) const

Definition at line 451 of file BaseClass.cpp.

```
00451                                             {
00452   if (i < op2.i ) return true;
00453   if (i == op2.i) {
00454     if (s <= op2.s) return true;
00455   }
00456   return false;
00457 }
```

4.17.3.4 bool operator== (const iskey & *op2*) const

Definition at line 417 of file BaseClass.cpp.

```
00417                                             {
00418   if(op2.i == i && op2.s == s){
00419     return true;
00420   }
00421   return false;
00422 }
```

4.17.3.5 bool operator> (const iskey & *op2*) const

Definition at line 442 of file BaseClass.cpp.

```
00442                                            {
00443   if (i > op2.i ) return true;
00444   if (i == op2.i) {
00445     if (s > op2.s) return true;
00446   }
00447   return false;
00448 }
```

4.17.3.6 bool operator>= (const iskey & *op2*) const

Definition at line 460 of file BaseClass.cpp.

```
00460                                       {
00461   if (i > op2.i ) return true;
00462   if (i == op2.i) {
00463     if (s >= op2.s) return true;
00464   }
00465   return false;
00466 }
```

4.17.4 Member Data Documentation

4.17.4.1 int i

Definition at line 187 of file BaseClass.h.

Referenced by operator!=(), operator<(), operator<=(), operator==(), operator>(), and operator>=().

4.17.4.2 string s

Definition at line 188 of file BaseClass.h.

Referenced by operator!=(), operator<(), operator<=(), operator==(), operator>(), and operator>=().

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

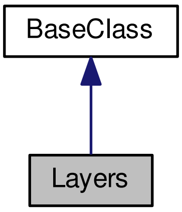
- /home/lobianco/git/ffsm_pp/src/BaseClass.h
- /home/lobianco/git/ffsm_pp/src/BaseClass.cpp

4.18 Layers Class Reference

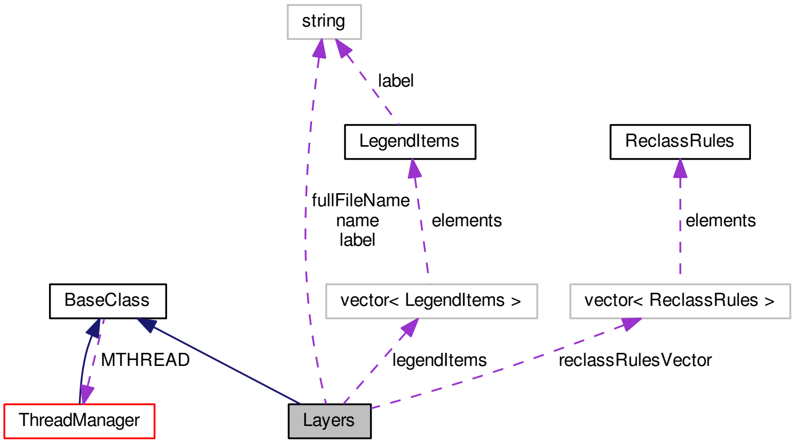
Define layer objects at the regional level.

`#include <Layers.h>`

Inheritance diagram for Layers:



Collaboration diagram for Layers:



Public Member Functions

- Layers (ThreadManager ∗MTHREAD_h, string name_h, string label_h, bool isInteger_h, bool dynamicContent_h, string fullFilename_h, bool display_h=true)

  In the constructor we set the main layer properties.
- ∼Layers ()
- void addLegendItem (int ID_h, string label_h, int rColor_h, int gColor_h, int bColor_h, double minValue_h, double maxValue_h)

  Add a legend item.
- void addLegendItems (vector< LegendItems > legendItems_h)
- vector< LegendItems > getLegendItems ()
- QColor getColor (double ID_h)

  Evaluates all the legend items to find the one that match the input code, and return its color as a QColor.
- string getCategory (double ID_h)

  Evaluates all the legend items to find the one that match the input code, and return its label.
- double filterExogenousDataset (double code_h)

  Used to reclassify the land use map for "generic" categories.
- void countMyPixels (bool debug=false)

  Count the pixels going to each legend item and print them if debug==true.
- void randomShuffle ()

  For some sensitivity analisys, random the values for this layer for not-empty values (only integer layers)
- bool getIsInteger ()

  Return if the layer is integer or not (If integer on each legend item: minValue==maxValue==ID)
- void print ()

  Print the layer content as an ASCII grid map with its companion files (classification and colors). It always print the whole region, even when subregion is actived.
- void printBinMap ()

  Print a binary reppresentation of the data (a standard image, e.g. a .png file). It prints only the subregion if this is active.
- string getName () const
- string getFilename ()

  Return the filename of the associated dataset.

- bool getDynamicContent ()

  Return true if the content may change during simulation period.
- bool getDisplay ()

Private Attributes

- string name

  ID of the layer (no spaces allowed)
- string label

  Label of the layer (spaces allowed)
- bool isInteger

  Type of the layer (true==integer, false==double. If true, on each legend item: minValue==maxValue==ID)
- bool dynamicContent

  True if the content may change during simulation year.
- bool display

  Normally true, but some layers used to just keep data shoudn't be normally processed.
- string fullFileName

  Filename of the associated dataset (map)
- vector< LegendItems > legendItems

  Vector of legend items.
- vector< ReclassRules > reclassRulesVector

  Vector of initial reclassification rules.

Additional Inherited Members

4.18.1 Detailed Description

Define layer objects at the regional level.

Layer class (setting, legend...)
This class define layer objects, including:

- a set of layer proprieties (name(ID), label, associated dataset, typology (integer or double)
- a vector of legend items, associating one color to each value or interval
- a vector of reclassification rule, when we need to work with a level of depth different of those coming with the dataset

  Author

  Antonello Lobianco `antonello@regmas.org`

Definition at line 49 of file Layers.h.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 Layers (ThreadManager ∗ *MTHREAD_h*, string *name_h*, string *label_h*, bool *isInteger_h*, bool *dynamicContent_h*, string *fullFilename_h*, bool *display_h* = `true`)

In the constructor we set the main layer properties.

Definition at line 32 of file Layers.cpp.

```
00033 {
00034   MTHREAD=MTHREAD_h;
00035   name = name_h;
00036   label = label_h;
00037   isInteger = isInteger_h;
00038   dynamicContent = dynamicContent_h;
00039   fullFileName = fullFilename_h;
00040   display = display_h;
00041 }
```

4.18.2.2 ∼Layers ( )

Definition at line 43 of file Layers.cpp.

```
00044 {
00045 }
```

4.18.3 Member Function Documentation

4.18.3.1 void addLegendItem (int *ID_h*, string *label_h*, int *rColor_h*, int *gColor_h*, int *bColor_h*, double *minValue_h*, double *maxValue_h*)

Add a legend item.

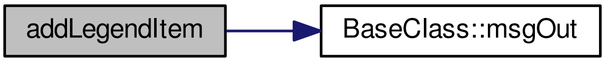
See also

LegendItems

Definition at line 48 of file Layers.cpp.

```
00048
                        {
00049
00050   for (uint i=0;i<legendItems.size();i++){
00051     if (legendItems.at(i).ID == ID_h){
00052       msgOut(MSG_ERROR, "Trying to add a legend item that already exist on this layer
       (layer: "+label+" - legend label: "+label_h+")");
00053       //cout << "ID: "<<ID_h<<" Label: "<<label_h<<" minValue: "<<minValue_h << " maxValue:
       "<<maxValue_h<<endl;
00054       return;
00055     }
00056   }
00057
00058   LegendItems ITEM;
00059   ITEM.ID = ID_h;
00060   ITEM.label = label_h;
00061   ITEM.rColor = rColor_h;
00062   ITEM.gColor = gColor_h;
00063   ITEM.bColor = bColor_h;
00064   ITEM.minValue = minValue_h;
00065   ITEM.maxValue = maxValue_h;
00066   ITEM.cashedCount=0;
00067   legendItems.push_back(ITEM);
00068
00069 }
```

Here is the call graph for this function:



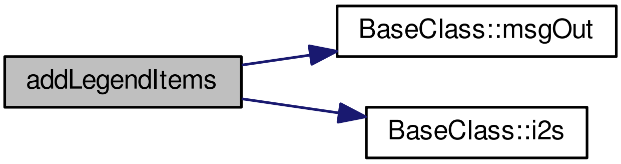
4.18.3.2 void addLegendItems (vector< LegendItems > *legendItems_h*)

Definition at line 72 of file Layers.cpp.

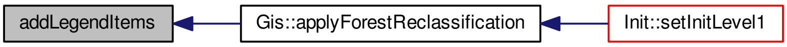
Referenced by Gis::applyForestReclassification().

```
00072                                                             {
00073   vector <LegendItems> toAdd;
00074   for(uint i=0; i<legendItems_h.size();i++){
00075     bool existing = false;
00076     for (uint j=0;j<legendItems.size();j++){
00077       if(legendItems_h[i].ID == legendItems[j].ID){
00078         existing = true;
00079         break;
00080       }
00081     }
00082     if(existing){
00083       msgOut(MSG_WARNING, "Legend item "+i2s(legendItems_h[i].ID)+" non added on layer
      "+this->name+" as already existing.");
00084     } else {
00085       toAdd.push_back(legendItems_h[i]);
00086     }
00087   }
00088   legendItems.insert( legendItems.end(), toAdd.begin(), toAdd.end() );
00089 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



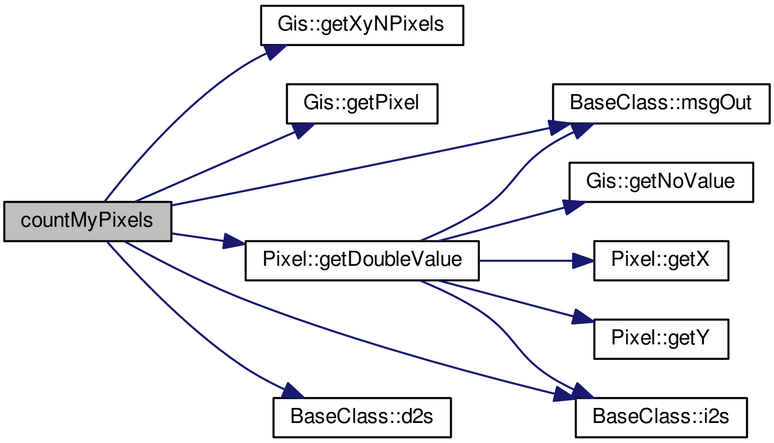
4.18.3.3 void countMyPixels (bool *debug* = `false`)

Count the pixels going to each legend item and print them if debug==true.

Definition at line 188 of file Layers.cpp.

```
00188                                  {
00189 
00190   for (uint i=0; i<legendItems.size(); i++){
00191     legendItems.at(i).cashedCount=0; //initialized with 0 values...
00192   }
00193   double totPixels = MTHREAD->GIS->getXyNPixels();
00194   double pixelValue;
00195   for (uint j=0;j<totPixels;j++){
00196     pixelValue = MTHREAD->GIS->getPixel(j)->getDoubleValue(
      name);
00197     if (isInteger){
00198       for(uint i=0; i<legendItems.size(); i++){
00199         if (legendItems.at(i).ID == ((int)pixelValue)){
00200           legendItems.at(i).cashedCount++;
00201           break;
00202         }
00203       }
00204     }
00205     else {
00206       for(uint i=0; i<legendItems.size(); i++){
00207         if (pixelValue < legendItems.at(i).maxValue &&  pixelValue >=
      legendItems.at(i).minValue){
00208           legendItems.at(i).cashedCount++;
00209           break;
00210         }
00211       }
00212     }
00213   }
00214   if (debug){
00215     msgOut(MSG_INFO, "Layer statistics - Count by Legend items");
00216     msgOut(MSG_INFO, "Layer name: "+label);
00217     msgOut(MSG_INFO, "Total plots: "+ d2s(totPixels));
00218     for(uint i=0;i<legendItems.size();i++){
00219       msgOut(MSG_INFO, legendItems.at(i).label+": "+i2s(
      legendItems.at(i).cashedCount));
00220     }
00221   }
00222 }
```

Here is the call graph for this function:



4.18.3.4 double filterExogenousDataset (double *code_h*)

Used to reclassify the land use map for "generic" categories.

Used in the init stage, this function take as input the real map code as just read from the map file, and filter it according to the reclassification rules.

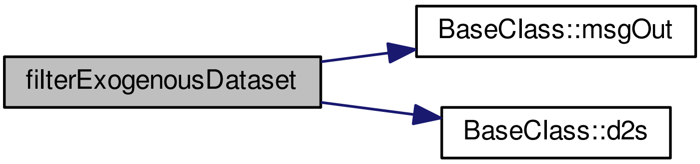
See also

ReclassRules

Definition at line 97 of file Layers.cpp.

```
00097                                              {
00098   bool check =false;
00099   std::vector <double> cumPVector;
00100   std::vector <double> outCodesVector;
00101   double cumP = 0;
00102   double returnCode=0;
00103
00104   for(uint i=0; i<reclassRulesVector.size(); i++){
00105     if (reclassRulesVector.at(i).inCode == code_h){
00106       check = true;
00107       cumP += reclassRulesVector.at(i).p;
00108       cumPVector.push_back(cumP);
00109       outCodesVector.push_back(reclassRulesVector.at(i).outCode);
00110     }
00111   }
00112   if (!check) {return code_h;}
00113   if (cumP <= 0.99999999 || cumP >= 1.00000001){msgOut(MSG_CRITICAL_ERROR,"the sum
       of land use reclassification rules is not 1 for at least one input code (input code:  "+
      d2s(code_h)+"; cumP: "+d2s(cumP)+")");}
00114   double random;
00115   //srand(time(NULL)); // this would re-initialise the random seed
00116   random = ((double)rand() / ((double)(RAND_MAX)+(double)(1)) );
00117   for(uint i=0; i<cumPVector.size(); i++){
00118     if (random <= cumPVector.at(i)){
00119       returnCode = outCodesVector.at(i);
00120       break;
00121     }
00122   }
00123   return returnCode;
00124 }
```

Here is the call graph for this function:



4.18.3.5 string getCategory (double *ID_h*)

Evaluates all the legend items to find the one that match the input code, and return its label.

This function take as input the value stored in the pixel for the specific layer, loops over the legend item and find the one that match it, returning its label.
If the layer is of type integer, the match is agains legendItem IDs, otherwise we compare the legendItem ranges.

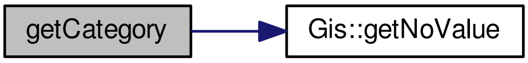
See also

LegendItems

Definition at line 162 of file Layers.cpp.

```
00162                                  {
00163   if (ID_h == MTHREAD->GIS->getNoValue()){
00164     return "";
00165   }
00166   if (isInteger){
00167     for(uint i=0; i<legendItems.size(); i++){
00168       if (legendItems.at(i).ID == ((int)ID_h)){
00169         return legendItems.at(i).label;
00170       }
00171     }
00172     return "";
00173   }
00174   else {
00175     for(uint i=0; i<legendItems.size(); i++){
00176       if (ID_h < legendItems.at(i).maxValue &&  ID_h >= legendItems.at(i).minValue){
00177         return legendItems.at(i).label;
00178       }
00179     }
00180     return "";
00181   }
00182 }
```

Here is the call graph for this function:



4.18.3.6 QColor getColor (double *ID_h*)

Evaluates all the legend items to find the one that match the input code, and return its color as a QColor.

This function take as input the value stored in the pixel for the specific layer, loops over the legend item and find the one that match it, returning its color.
If the layer is of type integer, the match is agains legendItem IDs, otherwise we compare the legendItem ranges.

See also

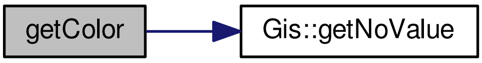
LegendItems

Definition at line 132 of file Layers.cpp.

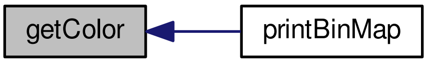
Referenced by printBinMap().

```
00132                           {
00133   QColor nocolor(255,255,255);
00134   if (ID_h == MTHREAD->GIS->getNoValue()){
00135     return nocolor;
00136   }
00137   if (isInteger){
00138     for(uint i=0; i<legendItems.size(); i++){
00139       if (legendItems.at(i).ID == ((int)ID_h)){
00140         QColor color(legendItems.at(i).rColor, legendItems.at(i).gColor,
      legendItems.at(i).bColor);
00141         return color;
00142       }
00143     }
00144     return nocolor;
00145   }
00146   else {
00147     for(uint i=0; i<legendItems.size(); i++){
00148       if (ID_h < legendItems.at(i).maxValue &&  ID_h >= legendItems.at(i).minValue){
00149         QColor color(legendItems.at(i).rColor, legendItems.at(i).gColor,
      legendItems.at(i).bColor);
00150         return color;
00151       }
00152     }
00153     return nocolor;
00154   }
00155 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.18.3.7 bool getDisplay ( ) [inline]

Definition at line 94 of file Layers.h.

```
00094 {return display;}
```

4.18.3.8 bool getDynamicContent ( ) [inline]

Return true if the content may change during simulation period.

Definition at line 93 of file Layers.h.

```
00093 {return dynamicContent;}
```

4.18.3.9 string getFilename ( ) [inline]

Return the filename of the associated dataset.

Definition at line 91 of file Layers.h.

```
00091 {return fullFileName;}
```

4.18.3.10 bool getIsInteger ( ) [inline]

Return if the layer is integer or not (If integer on each legend item: minValue==maxValue==ID)

Definition at line 83 of file Layers.h.

```
00083 {return isInteger;}
```

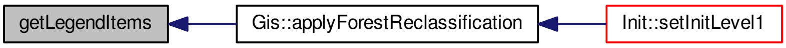
4.18.3.11 vector<LegendItems> getLegendItems ( ) [inline]

Definition at line 70 of file Layers.h.

Referenced by Gis::applyForestReclassification().

```
00070 {return legendItems;};
```

Here is the caller graph for this function:



4.18.3.12 string getName ( ) const [inline]

Definition at line 89 of file Layers.h.

```
00089 {return name;}
```

4.18.3.13 void print ( )

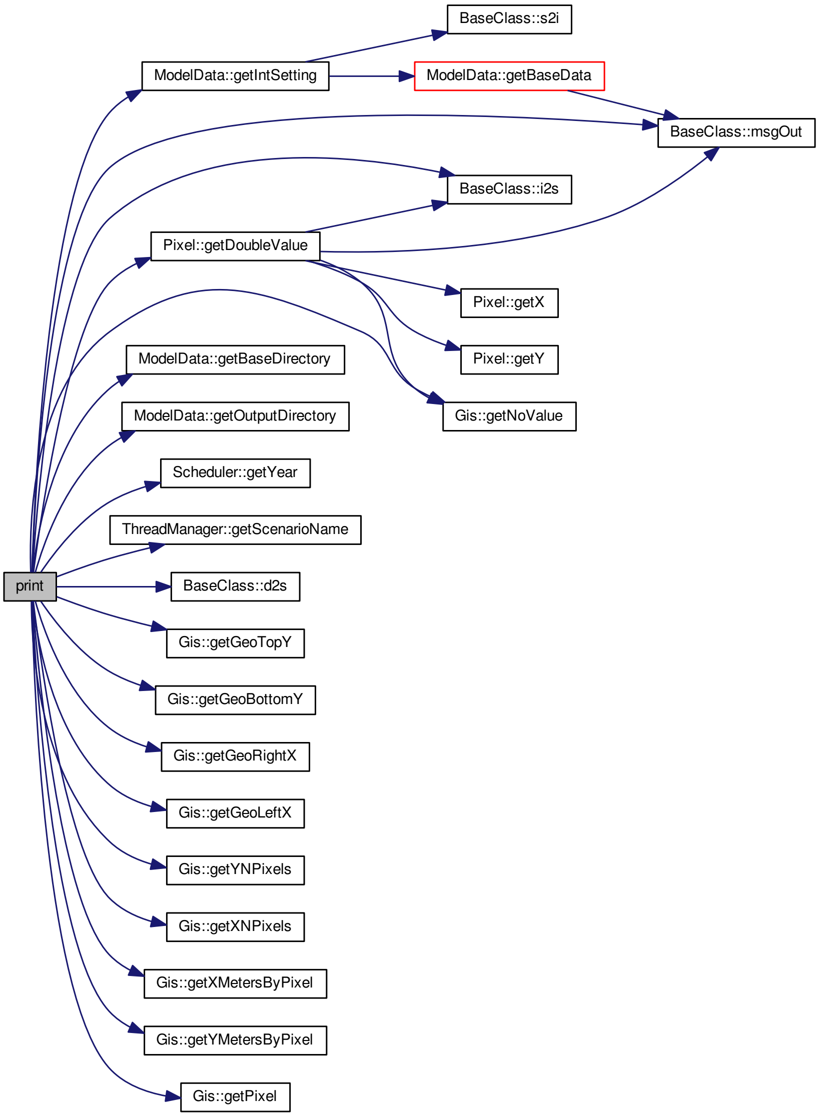
Print the layer content as an ASCII grid map with its companion files (classification and colors). It always print the whole region, even when subregion is actived.

Definition at line 251 of file Layers.cpp.

```
00251              {
00252
00253   if(MTHREAD->MD->getIntSetting("outputLevel")<OUTVL_MAPS) return;
00254   if(!display || !dynamicContent) return;
00255   string mapBaseDirectory = MTHREAD->MD->getBaseDirectory()+
      MTHREAD->MD->getOutputDirectory()+"maps/";
00256   string mapGridOutputDirectory = mapBaseDirectory+"asciiGrids/";
00257   string catsOutputDirectory = mapBaseDirectory+"cats/";
00258   string coloursOutputDirectory = mapBaseDirectory+"colr/";
00259
00260   string mapFilename      = mapGridOutputDirectory +name+ "_" +i2s(
      MTHREAD->SCD->getYear()) +"_" +MTHREAD->getScenarioName();
00261   string catsFilename     = catsOutputDirectory    +name+ "_" +i2s(
      MTHREAD->SCD->getYear()) +"_" +MTHREAD->getScenarioName();
00262   string coloursFilename  = coloursOutputDirectory +name+ "_" +i2s(
      MTHREAD->SCD->getYear()) +"_" +MTHREAD->getScenarioName();
00263     string filenameListIntLayers = mapBaseDirectory+"integerListLayers/"+MTHREAD->
      getScenarioName();
00264     string filenameListFloatLayers = mapBaseDirectory+"floatListLayers/"+MTHREAD->
      getScenarioName();
00265
00266   // printing the map...
00267   string header;
00268   if(MTHREAD->MD->getIntSetting("mapOutputFormat") == 1){ // GRASS ASCII Grid
00269     header =      "north: " + d2s(MTHREAD->GIS->getGeoTopY()) + "\n"
00270           + "south: " + d2s(MTHREAD->GIS->getGeoBottomY()) + "\n"
00271           + "east: " + d2s(MTHREAD->GIS->getGeoRightX()) + "\n"
00272           + "west: " + d2s(MTHREAD->GIS->getGeoLeftX()) + "\n"
00273           + "rows: " + i2s(MTHREAD->GIS->getYNPixels()) + "\n"
00274           + "cols: " + i2s(MTHREAD->GIS->getXNPixels()) + "\n"
00275           + "null: " + d2s(MTHREAD->GIS->getNoValue()) + "\n";
00276
00277   } else if(MTHREAD->MD->getIntSetting("mapOutputFormat") == 2){
00278     header =      "ncols: " + i2s(MTHREAD->GIS->getXNPixels()) + "\n"
00279           + "lrows: " + i2s(MTHREAD->GIS->getYNPixels()) + "\n"
00280           + "xllcornel: " + d2s(MTHREAD->GIS->getGeoLeftX()) + "\n"
00281           + "yllcorner: " + d2s(MTHREAD->GIS->getGeoBottomY()) + "\n"
00282           + "cellsize: "  + d2s(MTHREAD->GIS->getXMetersByPixel()) + "\n"
00283           + "nodata_value: " + d2s(MTHREAD->GIS->getNoValue()) + "\n";
00284     if(MTHREAD->GIS->getXMetersByPixel() != MTHREAD->
      GIS->getYMetersByPixel()){
00285       msgOut(MSG_ERROR, "The X resolution is different to the Y resolution. I am exporting
       the map in ArcInfo ASCII Grid format using the X resolution, but be aware that it is incorrect, as this
       format doesn't support different X-Y resolutions.");
00286     }
00287
00288   } else {
00289     msgOut(MSG_ERROR,"Map not print for unknow output type.");
00290   }
00291
00292   ofstream outm; //out map
00293   outm.open(mapFilename.c_str(), ios::out); //ios::app to append..
00294   if (!outm){ msgOut(MSG_ERROR,"Error in opening the file "+mapFilename+".");}
00295   outm << header << "\n";
00296
00297   for (int i=0;i<MTHREAD->GIS->getYNPixels();i++){
00298     for (int j=0;j<MTHREAD->GIS->getXNPixels();j++){
00299       outm << MTHREAD->GIS->getPixel(j, i)->getDoubleValue(
      name) << " ";
00300     }
00301     outm << "\n";
00302   }
00303   outm.close();
00304
00305   //printing the cat file
00306   ofstream outc; //out category file
00307   outc.open(catsFilename.c_str(), ios::out); //ios::app to append..
00308   if (!outc){ msgOut(MSG_ERROR,"Error in opening the file "+catsFilename+".");}
00309   outc << "# " << name  << "_-_" << i2s(MTHREAD->SCD->getYear()) << "\n\n\n";
00310   outc << "0.00 0.00 0.00 0.00"<<"\n";
00311
00312   if (isInteger){
00313     for(uint i=0;i<legendItems.size();i++){
00314       outc << legendItems[i].ID << ":"<< legendItems[i].label << "\n";
00315     }
00316   }
```

```
  else {
    for(uint i=0;i<legendItems.size();i++){
      outc << legendItems[i].minValue << ":"<< legendItems[i].maxValue << ":"<< legendItems[i].label << "\n";
    }
  }

  //printing the colour legend file
  ofstream outcl; //out colour file
  outcl.open(coloursFilename.c_str(), ios::out); //ios::app to append..
  if (!outcl){ msgOut(MSG_ERROR,"Error in opening the file "+coloursFilename+".");}
  outcl << "% " << name  << "_-_" << i2s(MTHREAD->SCD->getYear()) << "\n\n\n";

  if (isInteger){
    for(uint i=0;i<legendItems.size();i++){
      outcl << legendItems[i].ID << ":"<< legendItems[i].rColor << ":" << legendItems[i].gColor << ":" << legendItems[i].bColor << "\n";
    }
  }
  else {
    for(uint i=0;i<legendItems.size();i++){
      outcl << legendItems[i].minValue << ":"<< legendItems[i].rColor << ":" << legendItems[i].gColor << ":" << legendItems[i].bColor << " "<< legendItems[i].maxValue << ":"<< legendItems[i].rColor << ":" << legendItems[i].gColor << ":" << legendItems[i].bColor << "\n";
    }
  }

  // adding the layer to the list of saved layers..
  ofstream outList;
  if (isInteger){
    outList.open(filenameListIntLayers.c_str(), ios::app); // append !!!
    outList << name << "_" << MTHREAD->SCD->getYear() << "_" << MTHREAD->getScenarioName() << "\n";
  }
  else {
    outList.open(filenameListFloatLayers.c_str(), ios::app); // append !!!
    outList << name << "_" << MTHREAD->SCD->getYear() << "_" << MTHREAD->getScenarioName() << "\n";
  }
  outList.close();
}
```

Here is the call graph for this function:



4.18.3.14 void printBinMap ( )

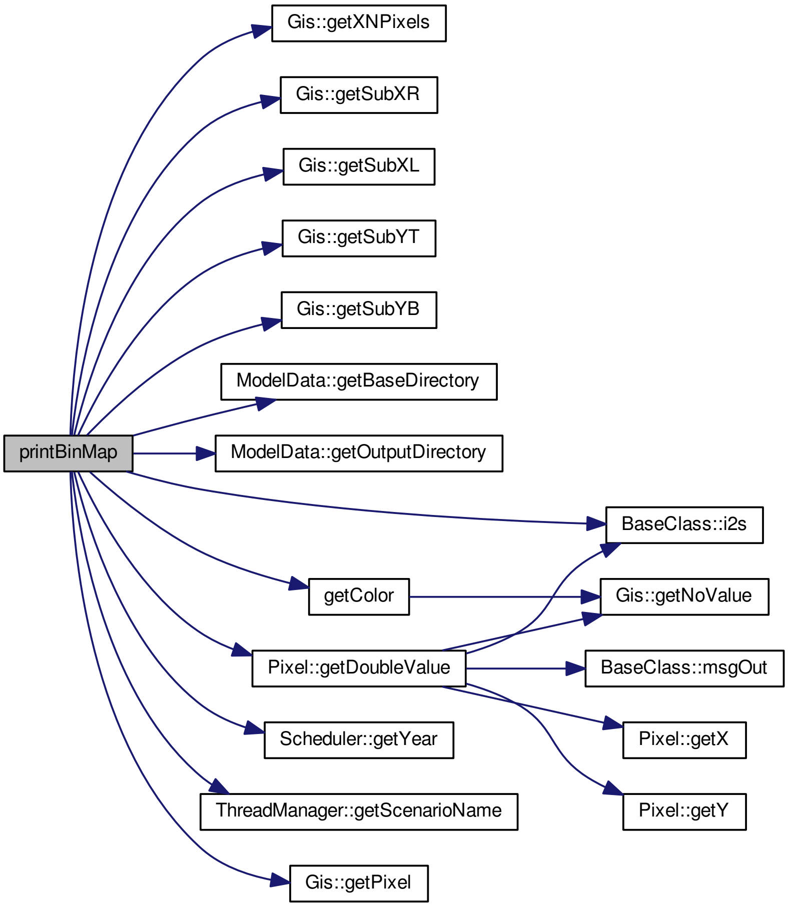
Print a binary reppresentation of the data (a standard image, e.g. a .png file). It prints only the subregion if this is active.

Definition at line 354 of file Layers.cpp.

```
00354                     {
00355 
00356   if(!display || !dynamicContent) return;
00357 
00358   int xNPixels            = MTHREAD->GIS->getXNPixels();
00359   int subXR               = MTHREAD->GIS->getSubXR();
00360   int subXL               = MTHREAD->GIS->getSubXL();
00361   int subYT               = MTHREAD->GIS->getSubYT();
00362   int subYB               = MTHREAD->GIS->getSubYB();
00363 
```

```
  string mapBaseDirectory = MTHREAD->MD->getBaseDirectory()+MTHREAD->MD->getOutputDirectory()+"maps/bitmaps/";
  string mapFilename      = mapBaseDirectory +name+ "_" +i2s(MTHREAD->SCD->getYear()) +"_" +MTHREAD->getScenarioName()+".png";

  QImage image = QImage(subXR-subXL+1, subYB-subYT+1, QImage::Format_RGB32);
  image.fill(qRgb(255, 255, 255));
  for (int countRow=subYT;countRow<subYB;countRow++){
    for (int countColumn=subXL;countColumn<subXR;countColumn++){
      double value = MTHREAD->GIS->getPixel(countRow*xNPixels+countColumn)->getDoubleValue(name);
      QColor color = this->getColor(value);
      image.setPixel(countColumn-subXL,countRow-subYT,color.rgb());
    }
  }
  image.save(mapFilename.c_str());
}
```

Here is the call graph for this function:



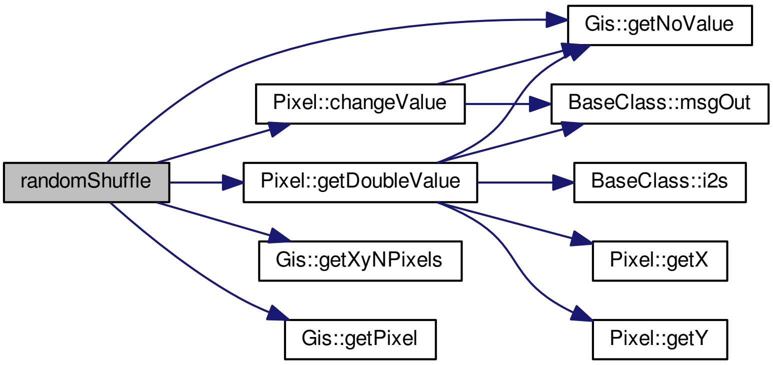
4.18.3.15 void randomShuffle ( )

For some sensitivity analisys, random the values for this layer for not-empty values (only integer layers)

Definition at line 224 of file Layers.cpp.

```
                     {


  vector <double> origValues;
  int maskValue = -MTHREAD->GIS->getNoValue();
  double totPixels = MTHREAD->GIS->getXyNPixels();
  for (uint i=0;i<totPixels;i++){
    double pxValue= MTHREAD->GIS->getPixel(i)->getDoubleValue(
name);
    if(pxValue != MTHREAD->GIS->getNoValue()){
      origValues.push_back(pxValue);
      MTHREAD->GIS->getPixel(i)->changeValue(name,maskValue);
    }
  }
  random_shuffle(origValues.begin(), origValues.end()); // randomize the elements of the array.

  for (uint i=0;i<totPixels;i++){
    double pxValue= MTHREAD->GIS->getPixel(i)->getDoubleValue(
name);
    if(pxValue != MTHREAD->GIS->getNoValue()){
      double toChangeValue = origValues.at(origValues.size()-1);
      //cout << toChangeValue << endl;
      origValues.pop_back();
      MTHREAD->GIS->getPixel(i)->changeValue(name,toChangeValue);
    }
  }

}
```

Here is the call graph for this function:



4.18.4 Member Data Documentation

4.18.4.1 bool display `[private]`

Normally true, but some layers used to just keep data shoudn't be normally processed.

Definition at line 102 of file Layers.h.

Referenced by Layers(), print(), and printBinMap().

4.18.4.2 bool dynamicContent `[private]`

True if the content may change during simulation year.

Definition at line 101 of file Layers.h.

Referenced by Layers(), print(), and printBinMap().

4.18.4.3 string fullFileName [private]

Filename of the associated dataset (map)

Definition at line 103 of file Layers.h.

Referenced by Layers().

4.18.4.4 bool isInteger [private]

Type of the layer (true==integer, false==double. If true, on each legend item: minValue==maxValue==ID)

Definition at line 100 of file Layers.h.

Referenced by countMyPixels(), getCategory(), getColor(), Layers(), and print().

4.18.4.5 string label [private]

Label of the layer (spaces allowed)

Definition at line 99 of file Layers.h.

Referenced by addLegendItem(), countMyPixels(), and Layers().

4.18.4.6 vector< LegendItems > legendItems [private]

Vector of legend items.

See also

LegendItems

Definition at line 104 of file Layers.h.

Referenced by addLegendItem(), addLegendItems(), countMyPixels(), getCategory(), getColor(), and print().

4.18.4.7 string name [private]

ID of the layer (no spaces allowed)

Definition at line 98 of file Layers.h.

Referenced by addLegendItems(), countMyPixels(), Layers(), print(), printBinMap(), and randomShuffle().

4.18.4.8 vector< ReclassRules > reclassRulesVector [private]

Vector of initial reclassification rules.

See also

ReclassRules

Definition at line 105 of file Layers.h.

Referenced by filterExogenousDataset().

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

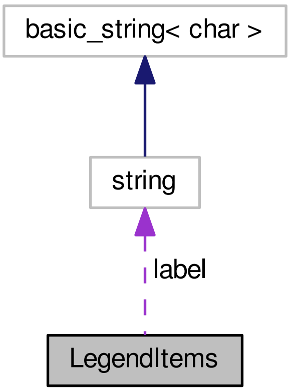
- /home/lobianco/git/ffsm_pp/src/Layers.h
- /home/lobianco/git/ffsm_pp/src/Layers.cpp

4.19 LegendItems Struct Reference

Legend items.

`#include <Layers.h>`

Collaboration diagram for LegendItems:



Public Attributes

- int ID
- string label
- int rColor
- int gColor
- int bColor
- double minValue
- double maxValue
- int cashedCount

    count the pixels whitin a item range

4.19.1 Detailed Description

Legend items.

Struct containing data about the programm settings.
The minValue and the maxValue are used to compare one record value and return the right color. If the layer is of type integer (isInteger==true), minValue==maxValue==ID.

Author

Antonello Lobianco

Definition at line 115 of file Layers.h.

4.19.2 Member Data Documentation

4.19.2.1 int bColor

Definition at line 120 of file Layers.h.

Referenced by Layers::addLegendItem().

4.19.2.2 int cashedCount

count the pixels whitin a item range

Definition at line 123 of file Layers.h.

Referenced by Layers::addLegendItem().

4.19.2.3 int gColor

Definition at line 119 of file Layers.h.

Referenced by Layers::addLegendItem().

4.19.2.4 int ID

Definition at line 116 of file Layers.h.

Referenced by Layers::addLegendItem().

4.19.2.5 string label

Definition at line 117 of file Layers.h.

Referenced by Layers::addLegendItem().

4.19.2.6 double maxValue

Definition at line 122 of file Layers.h.

Referenced by Layers::addLegendItem().

4.19.2.7 double minValue

Definition at line 121 of file Layers.h.

Referenced by Layers::addLegendItem().

4.19.2.8 int rColor

Definition at line 118 of file Layers.h.

Referenced by Layers::addLegendItem().

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

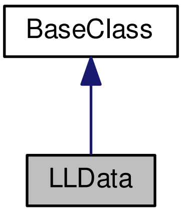
- /home/lobianco/git/ffsm_pp/src/Layers.h

4.20 LLData Class Reference

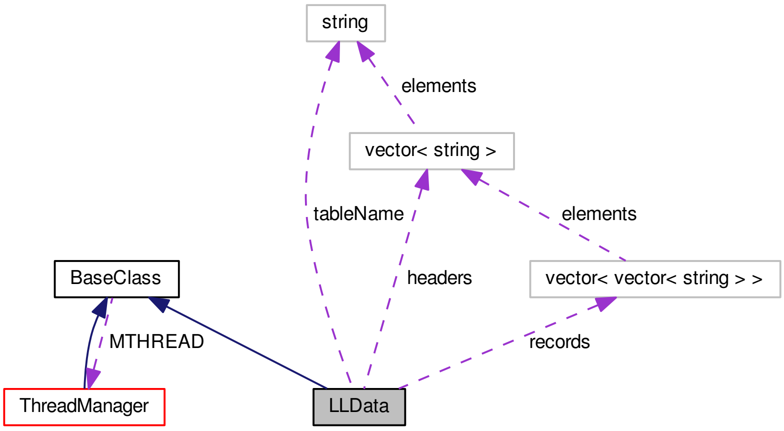
Low level data. XML input is reversed here after unzipping oocalc file and parsing content.xml.

`#include <ModelData.h>`

Inheritance diagram for LLData:



Collaboration diagram for LLData:



Public Member Functions

- LLData (ThreadManager ∗MTHREAD_h, string tableName_h)
- ∼LLData ()
- void clean ()
- string getTableName ()
- int nrecords ()
- int nheaders ()
- string getData (const int &pos_h, const string &header_h, const int &debugLevel=MSG_CRITICAL_ERROR) const

Private Attributes

- string tableName
- vector< string > headers
- vector< vector< string > > records

Friends

- void ModelData::loadInput ()
- void ModelData::loadDataFromCache (string tablename)

Additional Inherited Members

4.20.1 Detailed Description

Low level data. XML input is reversed here after unzipping oocalc file and parsing content.xml.

Definition at line 320 of file ModelData.h.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 LLData (ThreadManager ∗ *MTHREAD_h,* string *tableName_h*)

Definition at line 2009 of file ModelData.cpp.

```
02009                                                            {
02010   MTHREAD = MTHREAD_h;
02011   tableName = tableName_h;
02012 }
```

4.20.2.2 ∼LLData ( )

Definition at line 2014 of file ModelData.cpp.

```
02014                {
02015
02016 }
```

4.20.3 Member Function Documentation

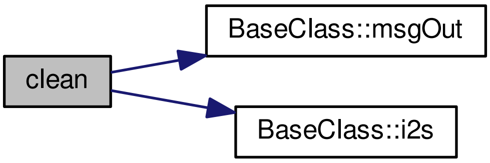
4.20.3.1 void clean ( )

Definition at line 2019 of file ModelData.cpp.

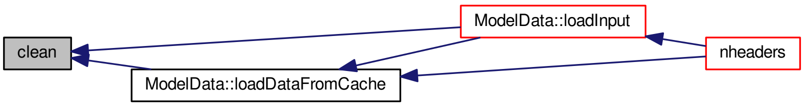
Referenced by ModelData::loadDataFromCache(), and ModelData::loadInput().

```
02019             {
02020 
02021   //checking the size is correct...
02022   int hsize = headers.size();
02023   for (uint i=0;i<records.size();i++){
02024     if(records[i].size() != hsize){
02025       vector <string> record = records[i];
02026       msgOut(MSG_CRITICAL_ERROR,"Error in the input reading table "+
      tableName+". Record "+i2s(i)+" has "+i2s(records[i].size())+" fields instead of "+
      i2s(hsize)+".");
02027     }
02028   }
02029   //cleaning empty-header columns...
02030   for (int i=headers.size()-1;i>=0;i--){
02031     if(headers[i] == ""){
02032       headers.erase(headers.begin()+i);
02033       for (uint j=0;j<records.size();j++){
02034         records[j].erase(records[j].begin()+i);
02035       }
02036     }
02037   }
02038 
02039 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



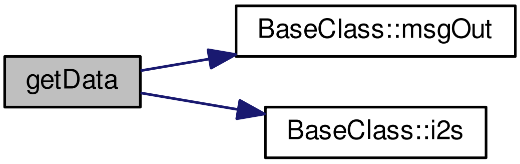
4.20.3.2 string getData (const int & *pos_h*, const string & *header_h*, const int & *debugLevel* = MSG_CRITICAL_ERROR) const

Definition at line 2042 of file ModelData.cpp.

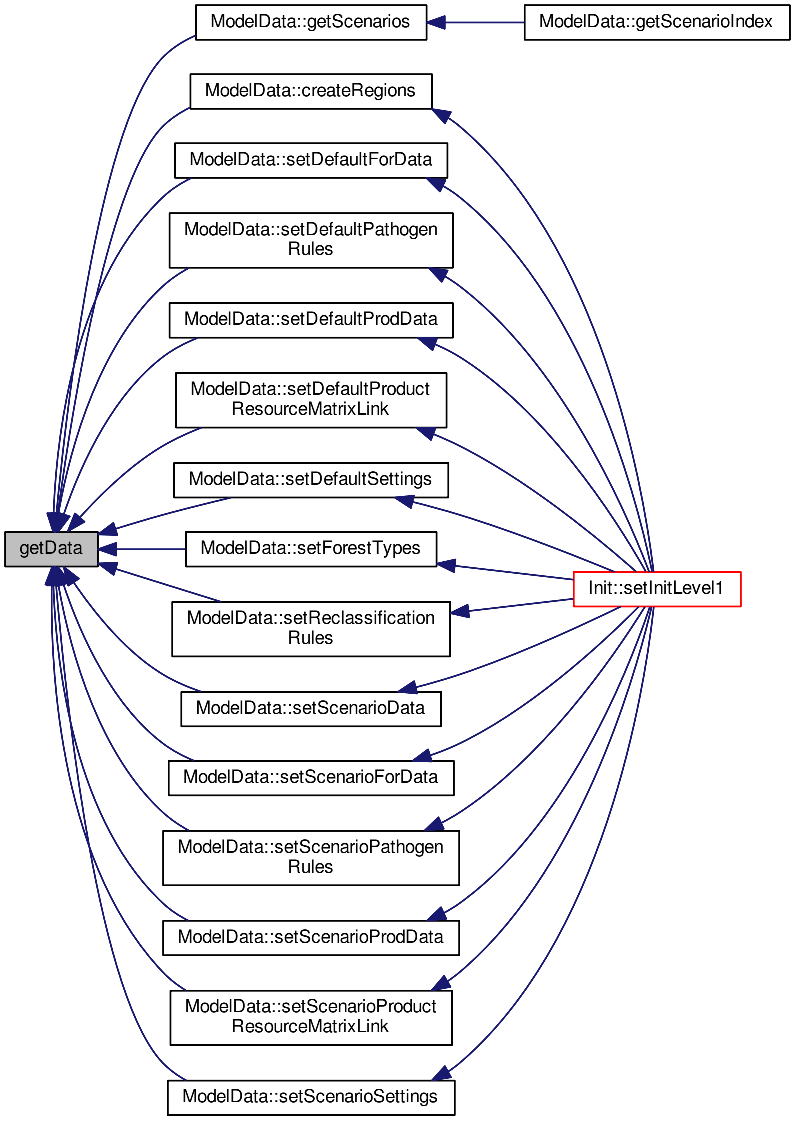
Referenced by ModelData::createRegions(), ModelData::getScenarios(), ModelData::setDefaultForData(), ModelData::setDefaultPathogenRules(), ModelData::setDefaultProdData(), ModelData::setDefaultProductResourceMatrixLink(), ModelData::setDefaultSettings(), ModelData::setForestTypes(), ModelData::setReclassificationRules(), ModelData::setScenarioData(), ModelData::setScenarioForData(), ModelData::setScenarioPathogenRules(), ModelData::setScenarioProdData(), ModelData::setScenarioProductResourceMatrixLink(), and ModelData::setScenarioSettings().

```
02042                                                                                    {
02043
02044   if (records.size()<= pos_h){
02045     msgOut(debugLevel, "Requested position "+i2s(pos_h)+" too high! Not enought records !!");
02046     return "";
02047   }
02048   int hsize = headers.size();
02049   for (uint i=0;i<hsize;i++){
02050     if(headers[i] == header_h) return records[pos_h][i];
02051   }
02052   msgOut(debugLevel, "Header string "+header_h+" not found!");
02053   return "";
02054 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


ModelData::getScenarios
ModelData::getScenarioIndex
ModelData::createRegions
ModelData::setDefaultForData
ModelData::setDefaultPathogen Rules
ModelData::setDefaultProdData
ModelData::setDefaultProduct ResourceMatrixLink
ModelData::setDefaultSettings
getData
ModelData::setForestTypes
Init::setInitLevel1
ModelData::setReclassification Rules
ModelData::setScenarioData
ModelData::setScenarioForData
ModelData::setScenarioPathogen Rules
ModelData::setScenarioProdData
ModelData::setScenarioProduct ResourceMatrixLink
ModelData::setScenarioSettings

4.20.3.3 string getTableName ( ) [inline]

Definition at line 326 of file ModelData.h.

```
00326 {return tableName;}
```

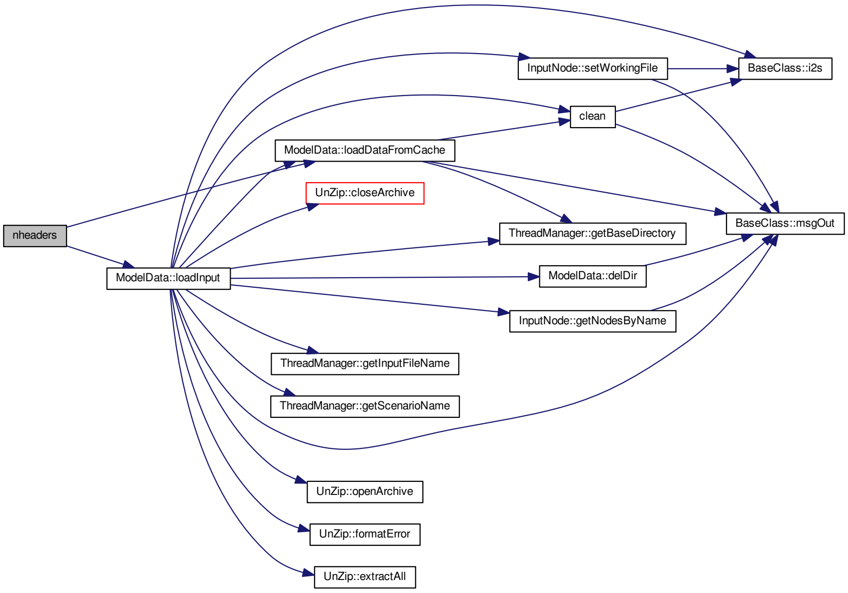
4.20.3.4 int nheaders ( ) [inline]

Definition at line 328 of file ModelData.h.

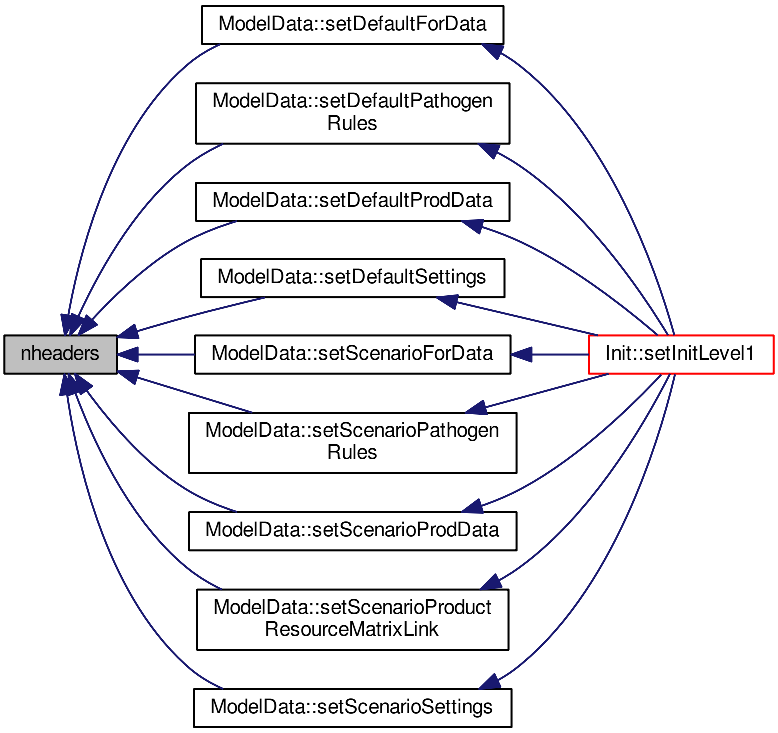
Referenced by ModelData::setDefaultForData(), ModelData::setDefaultPathogenRules(), ModelData::setDefaultProdData(), ModelData::setDefaultSettings(), ModelData::setScenarioForData(), ModelData::setScenarioPathogenRules(), ModelData::setScenarioProdData(), ModelData::setScenarioProductResourceMatrixLink(), and ModelData::setScenarioSettings().

```
00328 {return headers.size();}
```

Here is the call graph for this function:


nheaders
ModelData::loadInput
ModelData::loadDataFromCache
InputNode::setWorkingFile
BaseClass::i2s
clean
UnZip::closeArchive
ThreadManager::getBaseDirectory
BaseClass::msgOut
ModelData::delDir
InputNode::getNodesByName
ThreadManager::getInputFileName
ThreadManager::getScenarioName
UnZip::openArchive
UnZip::formatError
UnZip::extractAll

Here is the caller graph for this function:


ModelData::setDefaultForData
ModelData::setDefaultPathogen Rules
ModelData::setDefaultProdData
ModelData::setDefaultSettings
nheaders
ModelData::setScenarioForData
Init::setInitLevel1
ModelData::setScenarioPathogen Rules
ModelData::setScenarioProdData
ModelData::setScenarioProduct ResourceMatrixLink
ModelData::setScenarioSettings

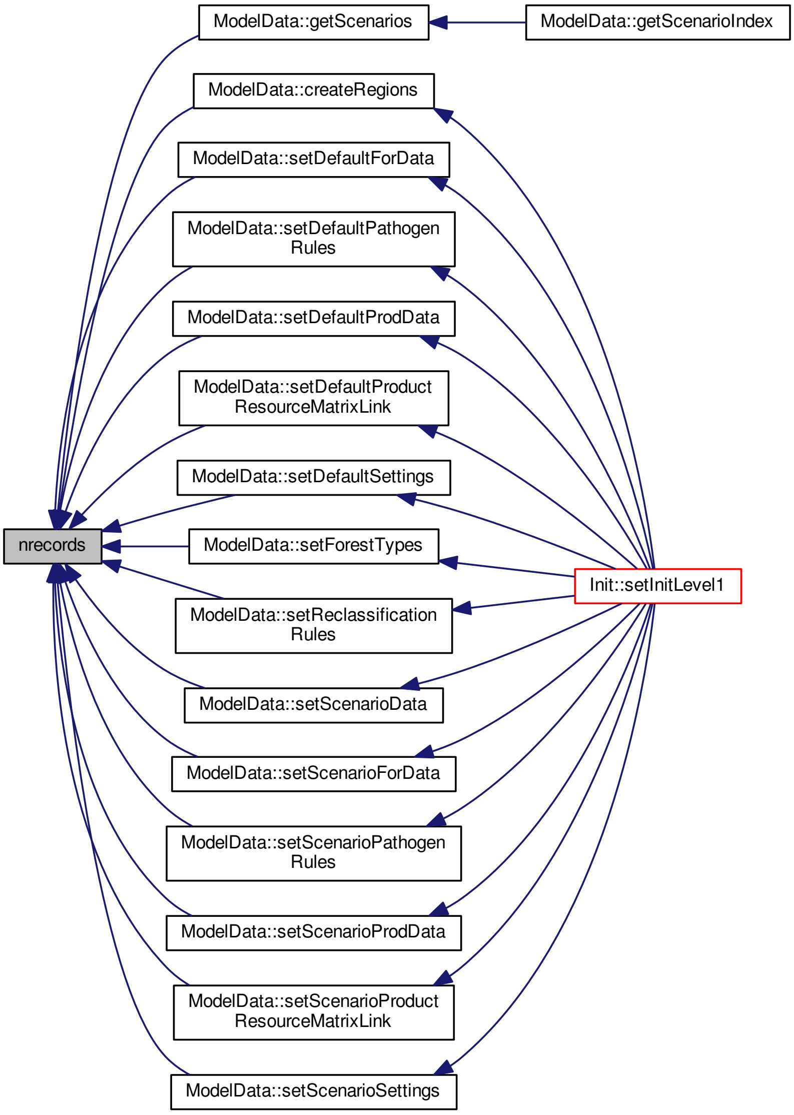
4.20.3.5 int nrecords ( ) [inline]

Definition at line 327 of file ModelData.h.

Referenced by ModelData::createRegions(), ModelData::getScenarios(), ModelData::setDefaultForData(), ModelData::setDefaultPathogenRules(), ModelData::setDefaultProdData(), ModelData::setDefaultProductResourceMatrixLink(), ModelData::setDefaultSettings(), ModelData::setForestTypes(), ModelData::setReclassificationRules(), ModelData::setScenarioData(), ModelData::setScenarioForData(), ModelData::setScenarioPathogenRules(), ModelData::setScenarioProdData(), ModelData::setScenarioProductResourceMatrixLink(), and ModelData::setScenarioSettings().

```
00327 {return records.size();}
```

Here is the caller graph for this function:


ModelData::getScenarios
ModelData::getScenarioIndex
ModelData::createRegions
ModelData::setDefaultForData
ModelData::setDefaultPathogen Rules
ModelData::setDefaultProdData
ModelData::setDefaultProduct ResourceMatrixLink
ModelData::setDefaultSettings
nrecords
ModelData::setForestTypes
Init::setInitLevel1
ModelData::setReclassification Rules
ModelData::setScenarioData
ModelData::setScenarioForData
ModelData::setScenarioPathogen Rules
ModelData::setScenarioProdData
ModelData::setScenarioProduct ResourceMatrixLink
ModelData::setScenarioSettings

4.20.4 Friends And Related Function Documentation

4.20.4.1 void **ModelData::loadDataFromCache** (string *tablename*) `[friend]`

4.20.4.2 void **ModelData::loadInput** ( ) `[friend]`

4.20.5 Member Data Documentation

4.20.5.1 vector<string> headers `[private]`

Definition at line 335 of file ModelData.h.

Referenced by ModelData::loadDataFromCache(), and ModelData::loadInput().

4.20.5.2 vector< vector <string> > records `[private]`

Definition at line 336 of file ModelData.h.

Referenced by ModelData::loadDataFromCache(), and ModelData::loadInput().

4.20.5.3 string tableName `[private]`

Definition at line 334 of file ModelData.h.

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

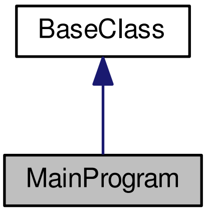
- /home/lobianco/git/ffsm_pp/src/ModelData.h
- /home/lobianco/git/ffsm_pp/src/ModelData.cpp

4.21 MainProgram Class Reference

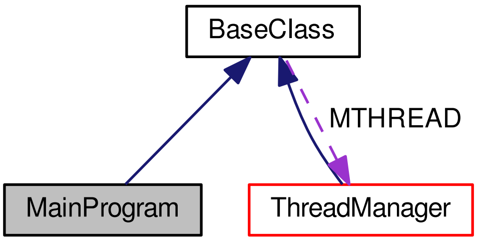
Main program scheleton. It control the flow of the program.

`#include <MainProgram.h>`

Inheritance diagram for MainProgram:



Collaboration diagram for MainProgram:



Public Member Functions

- MainProgram (ThreadManager ∗MTHREAD)
- ∼MainProgram ()
- void run ()

  Run the program.

Additional Inherited Members

4.21.1 Detailed Description

Main program scheleton. It control the flow of the program.

There is only one istance of this class. It is responsable to load the setting files, call the Init class, "speack" with the Scheduler and finally end the program.

Author

Antonello Lobianco

Definition at line 47 of file MainProgram.h.

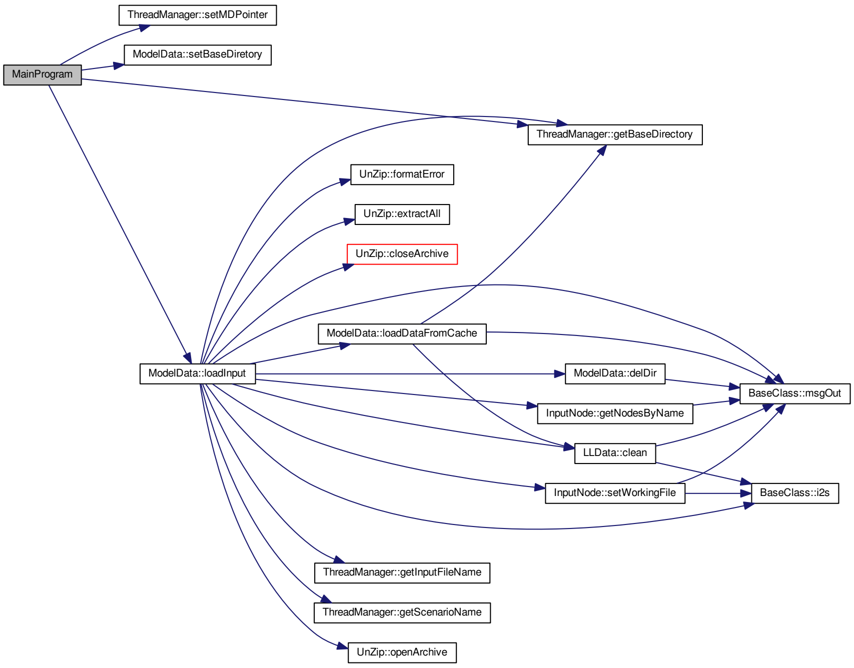
4.21.2 Constructor & Destructor Documentation

4.21.2.1 MainProgram (ThreadManager ∗ *MTHREAD*)

Definition at line 33 of file MainProgram.cpp.

```
00034 {
00035   //input_filename = input_filename_h;
00036   MTHREAD = MTHREAD_h;
00037   // Creating objects for the program flow:
00038   // the regional data object..
00039   ModelData *MD    = new ModelData(MTHREAD);
00040   MTHREAD->setMDPointer(MD);
00041   MTHREAD->MD->setBaseDiretory(MTHREAD->getBaseDirectory());
00042   MTHREAD->MD->loadInput(); // Unzip the ooffice input file and load it into memory
00043
00044 }
```

Here is the call graph for this function:


ThreadManager::setMDPointer
ModelData::setBaseDiretory
MainProgram
ThreadManager::getBaseDirectory
UnZip::formatError
UnZip::extractAll
UnZip::closeArchive
ModelData::loadDataFromCache
ModelData::loadInput
ModelData::delDir
BaseClass::msgOut
InputNode::getNodesByName
LLData::clean
InputNode::setWorkingFile
BaseClass::i2s
ThreadManager::getInputFileName
ThreadManager::getScenarioName
UnZip::openArchive

4.21.2.2 ∼MainProgram ( )

Definition at line 47 of file MainProgram.cpp.

```
00047                           {
00048 
00049 }
```

4.21.3 Member Function Documentation

4.21.3.1 void run ( )

Run the program.

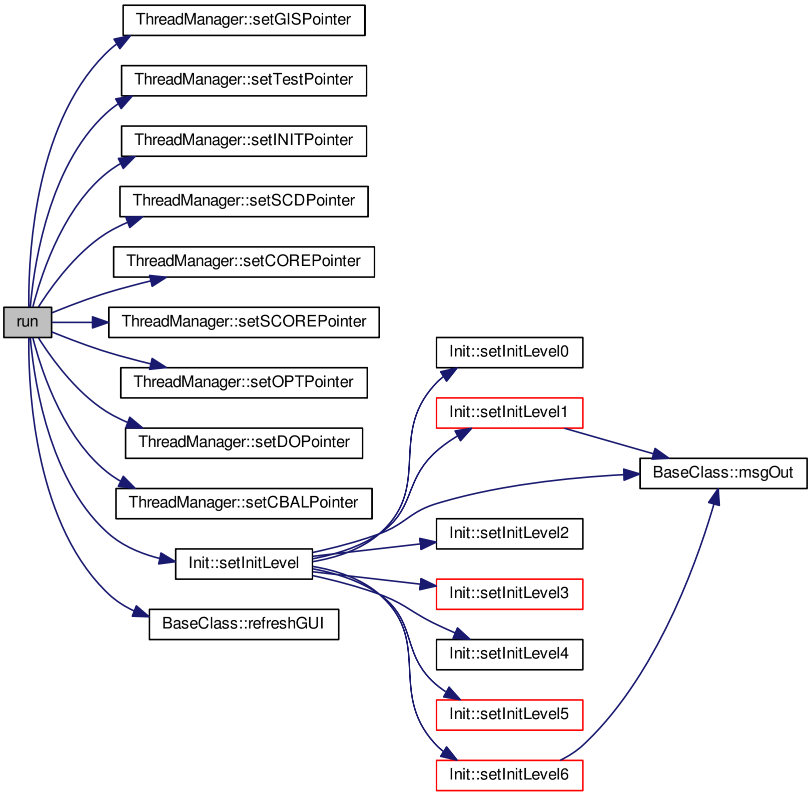
This is the main call of the program.
It firstly create the objects (and keep track of them trough pointers) of the main functional objects of the program.
Then it call the INIT object to do its jobs and when it ends, it gives control to SCD (Scheduler) for the year loops.
Finally it clean-up and returns.

Definition at line 58 of file MainProgram.cpp.

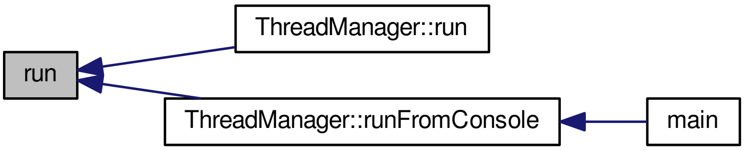
Referenced by ThreadManager::run(), and ThreadManager::runFromConsole().

```
                {

  setlocale(LC_ALL, "C"); // force to use the dot as digital separator also if we are running under the GUI

  // GIS information and methods..
  Gis *GIS  = new Gis(MTHREAD);
  MTHREAD->setGISPointer(GIS);
  // a test object for various 0-effects tests (sandbox)..
  Sandbox* TEST = new Sandbox(MTHREAD);
  MTHREAD->setTestPointer(TEST);
  // the Init object, it schedule the pre-simulation phase..
  Init *INIT = new Init(MTHREAD);
  MTHREAD->setINITPointer(INIT);
  // the scheduler object. It manage the simulation loops..
  Scheduler *SCD  = new Scheduler(MTHREAD);
  MTHREAD->setSCDPointer(SCD);
  // the core of the model
  ModelCore *CORE = new ModelCore(MTHREAD);
  MTHREAD->setCOREPointer(CORE);
  // the core of the model (spatial version)
  ModelCoreSpatial *SCORE = new ModelCoreSpatial(MTHREAD);
  MTHREAD->setSCOREPointer(SCORE);
  // the market optimisation algorithm
  Opt *OPT = new Opt(MTHREAD);
  MTHREAD->setOPTPointer(OPT);
  // manage the printing of data needed for scenario-analisys. The "message output" (needed to see "what is it happening?" are instead simply printed with msgOut()..
  Output *DO = new Output(MTHREAD);
  MTHREAD->setDOPointer(DO);
  // the carbon balance
  Carbon *CBAL = new Carbon(MTHREAD);
  MTHREAD->setCBALPointer(CBAL);

  // Creating an istance of INIT and delegating to it the Initialization phase..
  MTHREAD->INIT->setInitLevel(1); // Initial environment setting and agent rising
  refreshGUI();
  MTHREAD->INIT->setInitLevel(3); // assigning resources to agents and evenutal env reallocation
  refreshGUI();
  MTHREAD->INIT->setInitLevel(5); // starting simulations. Once INIT has ended it is the turn of SCD (Scheduler) to manage the simulation...
  refreshGUI();
  MTHREAD->INIT->setInitLevel(6); // ending simulations
  refreshGUI();

  // Deleting the pointers...
  // 20070102: if I delete the pointers I can not access the legend after simulation has ended
  // 20070109: pointers (e.g. INIT) are deleted in ThreadManager when a new simulation start
}
```

Here is the call graph for this function:


ThreadManager::setGISPointer
ThreadManager::setTestPointer
ThreadManager::setINITPointer
ThreadManager::setSCDPointer
ThreadManager::setCOREPointer
run
ThreadManager::setSCOREPointer
ThreadManager::setOPTPointer
ThreadManager::setDOPointer
ThreadManager::setCBALPointer
Init::setInitLevel
BaseClass::refreshGUI
Init::setInitLevel0
Init::setInitLevel1
BaseClass::msgOut
Init::setInitLevel2
Init::setInitLevel3
Init::setInitLevel4
Init::setInitLevel5
Init::setInitLevel6

Here is the caller graph for this function:



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

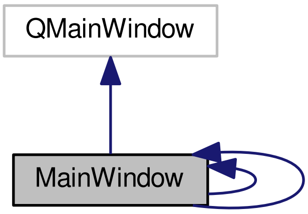
- /home/lobianco/git/ffsm_pp/src/MainProgram.h
- /home/lobianco/git/ffsm_pp/src/MainProgram.cpp

4.22 MainWindow Class Reference

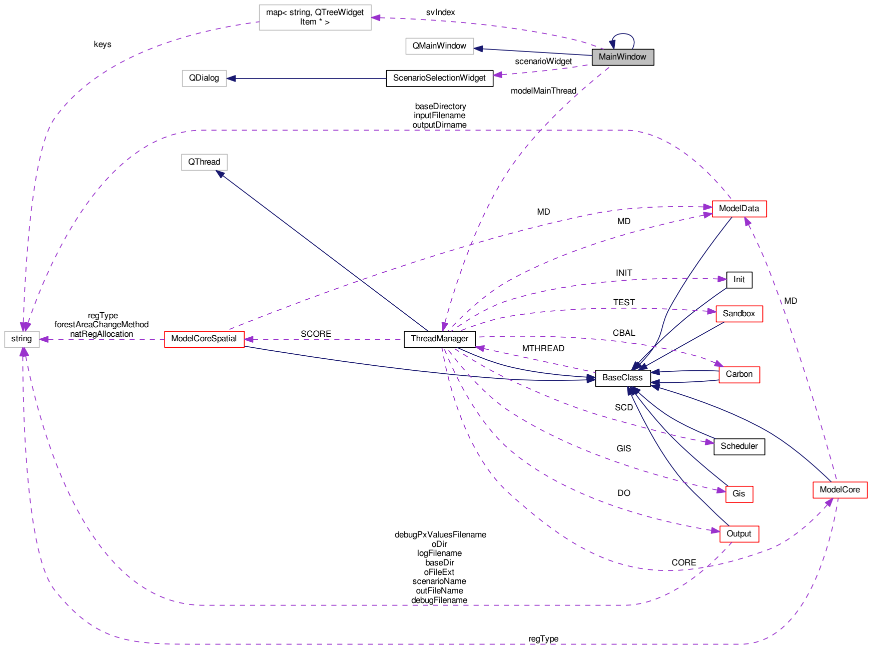
Main GUI interface.

`#include <MainWindow.h>`

Inheritance diagram for MainWindow:



Collaboration diagram for MainWindow:



Public Slots

- void setUnsavedStatus (bool unsavedStatus_h)

Signals

- void currentModelFilenameChanged (QString)
- void selectedScenarioName (const QString &scenarioName_h)
- void resized ()

Public Member Functions

- MainWindow ()

  Constructor.
- void setCurrentLogFileName (const QString &fileName)
- void setCurrentModelFileName (const QString &fileName)
- bool saveLogFile (const QString &logFileName)
- QString strippedName (const QString &fullFileName)
- QString getModelFileName ()
- void setModelFileName (const QString curModelFileName_h)
- void setOutputDirName (string outputDirName_h)
- void addLayer (QString layerName_h, QString layerLabel_h)
- void switchToLayer (QString layerName_h)
- void updatePixel (QString layerName_h, int x_h, int y_h, QColor color_h)
- void updateImage (QString layerName_h, const QImage &image_h)
- void switchToLayerFromLayerSelector (int layerIndex_h)
- void treeViewerItemChangeValue (string itemID, string newValue)

  Change value to an existing item in the Status Viewer.
- void treeViewerItemRemove (string itemID)
- void treeViewerAddItem (string text, string itemID, string parentID)

  e.g. manager_farmer_manager agents or agent_12345_ownedHa
- void processLogArea (const QString &message_h)
- void resetGUIForNewSimulation ()

  Reset the graphical elements for a new simulation // Send the request of getting the pixel info to the main thread.
- void receiveScenarioOptions (const QVector< QString > &scenarios_h)

Protected Member Functions

- void closeEvent (QCloseEvent ∗event)

  Manage the event of closing the application.
- void resizeEvent (QResizeEvent ∗event)

  Manage the event of resizing the application.

Private Types

- enum { MaxRecentFiles = 5 }

Private Slots

- void open ()
- bool save ()
- bool saveAs ()
- void startModelMainThread ()
- void stopModelMainThread ()
- void pauseOrResumeModelMainThread ()
- void openRecentFile ()
- void hideDebugMsgs (bool hide)
- void about ()
- void showDocumentation ()
- void openResults ()

Private Member Functions

- void createStatusBar ()
- bool okToContinue ()
- void readSettings ()
- void writeSettings ()
- void updateRecentFileActions ()

Private Attributes

- ThreadManager modelMainThread
- QLabel ∗ yearSBLabel

  Status bar current year label.
- QLabel ∗ mainSBLabel

  Status bar main label.
- bool unsavedStatus
- QString outputDirName
- QString curLogFileName
- QString curModelFileName
- QString curBaseDirectory
- QStringList recentFiles
- QAction ∗ recentFileActions [MaxRecentFiles]
- QAction ∗ separatorAction
- bool debugMsgsEnable

  Allow debug messages to be show in the logArea.
- ScenarioSelectionWidget ∗ scenarioWidget
- map< string, QTreeWidgetItem ∗ > svIndex

  Map containing the ID and the pointers to the status viewer.

Additional Inherited Members

4.22.1 Detailed Description

Main GUI interface.

MainWindow derive from both the generic Qt QMainWindow and from Ui::MainWindow (the latter being the autmatically generated C++ code from QtDesigner).
It implements code and functionality that can not be done in the QtDesigner.

Definition at line 50 of file MainWindow.h.

4.22.2 Member Enumeration Documentation

4.22.2.1 anonymous enum `[private]`

Enumerator

MaxRecentFiles

Definition at line 116 of file MainWindow.h.

```
00116 { MaxRecentFiles = 5 };
```

4.22.3 Constructor & Destructor Documentation

4.22.3.1 MainWindow ( )

Constructor.

It setup the Gui from the QTDesiger autogenerated code and connect various GUI signal/slots

Definition at line 39 of file MainWindow.cpp.

```
00039                        {
00040   yearSBLabel=NULL;
00041   mainSBLabel=NULL;
00042   for (uint i=0;i<MaxRecentFiles;i++) recentFileActions[i] = NULL;
00043   separatorAction=NULL;
00044 
00045   setupUi(this);
00046   unsavedStatus=false;
00047   curModelFileName="data/ffsmInput.ods";
00048   curBaseDirectory = QApplication::applicationDirPath();
00049   curBaseDirectory.append("/data/");
00050   //curBaseDirectory = "data/";
00051   outputDirName="output/";
00052   setCurrentLogFileName("");
00053   createStatusBar();
00054   curLogFileName ="";
00055   debugMsgsEnable = true;
00056 
00057   for (int i = 0; i < MaxRecentFiles; ++i) {
00058     recentFileActions[i] = new QAction(this);
00059     recentFileActions[i]->setVisible(false);
00060     connect(recentFileActions[i], SIGNAL(triggered()), this, SLOT(
      openRecentFile()));
00061   }
00062 
00063   separatorAction = menuFile->addSeparator();
00064   for (int i = 0; i < MaxRecentFiles; ++i)
00065     menuFile->addAction(recentFileActions[i]);
00066   menuFile->addSeparator();
00067   menuFile->addAction(actionExit);
00068 
00069   readSettings();
00070   modelMainThread.setInputFileName(
      curModelFileName);
00071   //modelMainThread.setBaseDirectory(curBaseDirectory);
00072 
00073   // Status viewer....
00074   statusView->setColumnCount(2);
00075   statusView->setHeaderLabels(QStringList()<< tr ("Label") << tr ("Value"));
00076   statusView->clear();
00077   statusView->sortByColumn(0);
00078   statusView->setFocus(); //????
00079 
00080 
00081 
00082 
00083   /*
00084   DONE: statusView should be implemented like this:
00085 
00086   Model
00087     -> year
00088     -> total plots
00089     -> rented plots
00090     -> abandoned plots
00091   Managers
00092     -> Manager_farmer
00093       -> number of agents
00094   Agents
00095     Agent_0
00096       -> Type
00097       -> ID
00098       -> mould
00099       -> owned plots
00100       ...
00101     Agent_1
00102       -> Type
00103       -> ID
00104       -> mould
00105       -> owned plots
00106       ...
```

```
    ...
  */

  qRegisterMetaType<string>("string"); // allows string objects to be thread-safely queued within signal-slots comunications
  qRegisterMetaType<QString>("QString");
  qRegisterMetaType< QVector<QString> >("QVector<QString>");


  connect(actionRun, SIGNAL(triggered()), this, SLOT(startModelMainThread()));
  connect(actionPause, SIGNAL(triggered()), this, SLOT(pauseOrResumeModelMainThread()));
  connect(actionStop, SIGNAL(triggered()), this, SLOT(stopModelMainThread()));
  connect(actionExit, SIGNAL(triggered()), this, SLOT(close()));
  connect(actionSaveLog, SIGNAL(triggered()), this, SLOT(save()));
  connect(actionSaveLogAs, SIGNAL(triggered()), this, SLOT(saveAs()));
  connect(actionLoadConfiguration, SIGNAL(triggered()), this, SLOT(open()));
  connect(actionHideDebugMsgs, SIGNAL(triggered(bool)), this, SLOT(hideDebugMsgs(bool)));
  connect(actionAboutRegMAS, SIGNAL(triggered()), this, SLOT(about()));
  connect(actionRegMASDocumentation, SIGNAL(triggered()), this, SLOT(showDocumentation()));
  connect(actionFitMap, SIGNAL(triggered()), mapBox, SLOT(fitInWindow()));
  connect(this, SIGNAL(resized()),mapBox, SLOT(fitInWindow()));
  connect(viewResultsButton, SIGNAL(clicked()),this, SLOT(openResults()));

  connect(&modelMainThread, SIGNAL(upgradeLogArea(const QString&)), this, SLOT(processLogArea(const QString&)));
  connect(&modelMainThread, SIGNAL(addLayerToGui(QString, QString)), this, SLOT(addLayer(QString, QString)));
  connect(layerSelector, SIGNAL(activated(int)), this, SLOT(switchToLayerFromLayerSelector(int)));
  connect(&modelMainThread, SIGNAL(updatePixelToGui(QString, int, int, QColor)), this, SLOT(updatePixel(QString, int, int, QColor)));
  connect(&modelMainThread, SIGNAL(updateImageToGui(QString, QImage)), this, SLOT (updateImage(QString, QImage)));
  connect(&modelMainThread, SIGNAL(setOutputDirNameToGui(string)), this, SLOT(setOutputDirName(string)));
  connect(&modelMainThread, SIGNAL(setGUIUnsavedStatus(bool)), this, SLOT(setUnsavedStatus(bool)));
  connect(&modelMainThread, SIGNAL(sendScenarioOptionsToGUI(const QVector<QString> &)), this, SLOT( receiveScenarioOptions(const QVector<QString> &)   ));

  // Scenario selection widget...
  scenarioWidget = new ScenarioSelectionWidget(this);
  connect(scenarioWidget->scenarioSelector, SIGNAL( activated(const QString&)), scenarioWidget, SLOT( close()));
  connect(scenarioWidget->scenarioSelector, SIGNAL( activated(const QString&)), &modelMainThread, SLOT( retrieveScenarioNameFromGUI(const QString &)));
  //connect(scenarioWidget, SIGNAL( selectedScenarioName(const QString&)), scenarioWidget, SLOT( close()));
  //connect(scenarioWidget, SIGNAL( selectedScenarioName(const QString&)), &modelMainThread, SLOT( retrieveScenarioNameFromGUI(const QString &)));

  // Model tree viewer...
  connect(&modelMainThread, SIGNAL( treeViewerItemChangeValueToGui(string, string)  ), this, SLOT( treeViewerItemChangeValue(string, string) ));
  connect(&modelMainThread, SIGNAL( treeViewerItemRemoveToGui(string)  ), this, SLOT( treeViewerItemRemove(string) ));
  connect(&modelMainThread, SIGNAL( treeViewerAddItemToGui(string, string, string)  ), this, SLOT( treeViewerAddItem(string, string, string) ));
  connect(&modelMainThread, SIGNAL( fitInWindowToGui()), mapBox, SLOT(fitInWindow()));

  connect(mapBox, SIGNAL(  queryRequestOnPx(int, int, bool) ), &modelMainThread, SLOT ( checkQuery(int, int, bool) ) );
  connect(&modelMainThread,SIGNAL(publishQueryResults(const QString&)), pxInfoArea, SLOT (setHtml(const QString&)));
  connect(&modelMainThread,SIGNAL(activateTab(int)), tabWidget, SLOT (setCurrentIndex(int)));

  connect(&modelMainThread, SIGNAL( resetGUIForNewSimulation()  ), this, SLOT( resetGUIForNewSimulation() ));

}
```

4.22.4 Member Function Documentation

4.22.4.1 void about ( ) [private], [slot]

Definition at line 570 of file MainWindow.cpp.

```
00570                  {
00571   QMessageBox::about(this, tr("About FFSM"),
00572     tr("<h2>FFSM</h2>"
00573        "<p>Copyright &copy; 2012 Laboratoire d’Economie Forestière - LEF"
00574        "<br/>"
00575        "<p>FFSM is a flexible, spatially explicit, coupled resource and economic simulator of the Forest Sector, "
00576        "designed for long-term simulations of effects of government policies "
00577        "over different forest systems."
00578        "<br>It is released under the GNU GPL licence."
00579        "<p>For documentation and credits please refer to the project site:"
00580        "<br><a href=\"http://www.ffsm-project.org\">http://www.ffsm-project.org</a>"
00581     ));
00582 }
```

4.22.4.2 void addLayer (QString *layerName_h,* QString *layerLabel_h*)

Perform all the operation needed when adding a new layer:

- add a layer to mapBox;
- add the layer to layerSelector;
- (NOTNEEDED: add the layer to layerLegend); Not needed any longer, as legend was dropped in name of the Model Status Viewer

Definition at line 440 of file MainWindow.cpp.

```
00440                                                               {
00441   static int counter =0;
00442   mapBox->addLayer(layerName_h);
00443   layerSelector->addItem(layerLabel_h,layerName_h);
00444   // first layer added only. it is not needed as MapBox::addLayer() and QComboBox automatically switch to the new value if it is the first one :-))
00445   //if (counter == 0) switchToLayer(layerName_h);
00446   update();
00447   counter ++;
00448 }
```

4.22.4.3 void closeEvent (QCloseEvent ∗ *event*) [protected]

Manage the event of closing the application.

Definition at line 181 of file MainWindow.cpp.

```
00181                                               {
00182   if (okToContinue()) {
00183     writeSettings();
00184     modelMainThread.stop();
00185     modelMainThread.wait();
00186     event->accept();
00187   } else {
00188     event->ignore();
00189     }
00190 }
```

4.22.4.4 void createStatusBar ( ) [private]

Definition at line 160 of file MainWindow.cpp.

```
00160                               {
00161   yearSBLabel = new QLabel(" 2000 ");
00162   yearSBLabel->setAlignment(Qt::AlignHCenter);
00163   yearSBLabel->setMinimumSize(yearSBLabel->sizeHint());
00164
00165   mainSBLabel = new QLabel;
00166   mainSBLabel->setIndent(3);
00167
00168   statusBar()->addWidget(yearSBLabel);
00169   statusBar()->addWidget(mainSBLabel, 1);
00170
00171   yearSBLabel->setText("0");
00172   mainSBLabel->setText("Welcome to FFSM!");
00173
00174   connect(&modelMainThread, SIGNAL(upgradeYearSBLabelToGui(const QString&)),
      yearSBLabel, SLOT(setText(const QString&)));
00175   connect(&modelMainThread, SIGNAL(upgradeMainSBLabelToGui(const QString&)),
      mainSBLabel, SLOT(setText(const QString&)));
00176
00177 }
```

4.22.4.5 void currentModelFilenameChanged (QString) [signal]

4.22.4.6 QString getModelFileName ( ) [inline]

Definition at line 61 of file MainWindow.h.

```
00061 {return curModelFileName;};
```

4.22.4.7 void hideDebugMsgs (bool *hide*) [private], [slot]

Definition at line 564 of file MainWindow.cpp.

```
00564                                    {
00565   if(hide) debugMsgsEnable = false;
00566   else debugMsgsEnable = true;
00567 }
```

4.22.4.8 bool okToContinue ( ) [private]

Definition at line 251 of file MainWindow.cpp.

```
00251                           {
00252   if (modelMainThread.isRunning()) {
00253     int t = QMessageBox::warning(
00254       this,                                         // parent
00255       tr("FFSM"),                                  // title
00256       tr("The model is still running.\n"     // message
00257         "Do you want to stop it?"),
00258       QMessageBox::Yes | QMessageBox::Default,     // 1st button
00259       QMessageBox::Cancel | QMessageBox::Escape    // 3rd button
00260     );
00261     if (t == QMessageBox::Yes) {
00262       modelMainThread.stop();
00263       modelMainThread.wait();
00264     } else if (t == QMessageBox::Cancel) {
00265       return false;
00266     }
00267   }
00268
00269   if (unsavedStatus) {
```

```
00270     int r = QMessageBox::warning(
00271       this,                                       // parent
00272       tr("FFSM"),                               // title
00273       tr("The model log has not been saved.\n"    // message
00274         "Do you want to save it?"),
00275       QMessageBox::Yes ,                          // 1st button
00276       QMessageBox::No | QMessageBox::Default,     // 2nd button
00277       QMessageBox::Cancel | QMessageBox::Escape   // 3rd button
00278     );
00279     if (r == QMessageBox::Yes) {
00280       return save();
00281     } else if (r == QMessageBox::Cancel) {
00282       return false;
00283     }
00284   }
00285   return true;
00286 }
```

4.22.4.9 void open ( ) [private], [slot]

Definition at line 289 of file MainWindow.cpp.

```
00289                  {
00290   if (okToContinue()) {
00291     QString fileName = QFileDialog::getOpenFileName(
00292       this,
00293       tr("Load model file.."),
00294       "data/",
00295       tr("OpenDocument Spreadsheet (*.ods)\n" "All files (*.*)")
00296     );
00297     if (!fileName.isEmpty()){
00298       statusBar()->showMessage(tr("Loaded new FFSM model file"), 2000);
00299       setCurrentModelFileName(fileName);
00300       // getting the baseData path information...
00301       QFileInfo info(fileName);
00302       QString path;
00303       path = info.absolutePath();
00304       path = path+"/";
00305       curBaseDirectory = path;
00306       //modelMainThread.setBaseDirectory(curBaseDirectory);
00307     }
00308     }
00309 }
```

4.22.4.10 void openRecentFile ( ) [private], [slot]

Definition at line 319 of file MainWindow.cpp.

```
00319                            {
00320   if (okToContinue()) {
00321     QAction *action = qobject_cast<QAction *>(sender());
00322     if (action){
00323       curModelFileName=action->data().toString();
00324       setCurrentModelFileName(curModelFileName);
00325       // getting the baseData path information...
00326       QFileInfo info(curModelFileName);
00327       QString path;
00328       path = info.absolutePath();
00329       path = path+"/";
00330       curBaseDirectory = path;
00331       //modelMainThread.setBaseDirectory(curBaseDirectory);
00332     }
00333   }
00334 }
```

4.22.4.11 void openResults ( ) [private], [slot]

Definition at line 680 of file MainWindow.cpp.

```
00680                         {
00681   //QLabel *label = new QLabel("Hello World!");
00682   //label->show();
00683   //string aaa = curBaseDirectory.toStdString();
00684   //cout << "curBaseDirectory " << aaa << endl;
00685   //cout << "outputDirName: " << outputDirName.toStdString() << endl;
00686   QUrl resultsUrl(curBaseDirectory+outputDirName+"results/results.ods",
      QUrl::TolerantMode);
00687   QDesktopServices::openUrl(resultsUrl);
00688
00689 }
```

4.22.4.12 void pauseOrResumeModelMainThread ( ) [private], [slot]

Definition at line 416 of file MainWindow.cpp.

```
00416                                          {
00417   modelMainThread.pauseOrResume();
00418 }
```

4.22.4.13 void processLogArea (const QString & *message_h*)

Definition at line 552 of file MainWindow.cpp.

```
00552                                                   {
00553   if(debugMsgsEnable){
00554     logArea->append(message_h);
00555   }
00556   else {
00557     if( ! message_h.startsWith("*DEBUG")){
00558       logArea->append(message_h);
00559     }
00560   }
00561 }
```

4.22.4.14 void readSettings ( ) [private]

Definition at line 312 of file MainWindow.cpp.

```
00312                           {
00313   QSettings settings("LEF", "FFSM");
00314   recentFiles = settings.value("recentFiles").toStringList();
00315   updateRecentFileActions();
00316 }
```

4.22.4.15 void receiveScenarioOptions (const QVector< QString > & *scenarios_h*)

Definition at line 664 of file MainWindow.cpp.

```
00664                                                                     {
00665
00666   //for(uint i=0;i<scenarios_h.size();i++){
00667   //  cout << scenarios_h.at(i).toStdString() << endl;
00668   //} // stange.. it works like expected !!!!
00669
00670   scenarioWidget->receiveScenarioOptions(scenarios_h);
00671   scenarioWidget->show();
00672   scenarioWidget->scenarioSelector->setFocus();
00673   //scenarioWidget->scenarioSelector->grabMouse();
00674   //scenarioWidget->scenarioSelector->grabKeyboard();
00675
00676
00677 }
```

4.22.4.16 void resetGUIForNewSimulation ( )

Reset the graphical elements for a new simulation // Send the request of getting the pixel info to the main thread.

Definition at line 607 of file MainWindow.cpp.

```
00607                                       {
00608 
00609   static int simulationCounter = 0;
00610   //reset map <string, QTreeWidgetItem*>       svIndex and clean the tree widget
00611   statusView->clear();
00612   map<string, QTreeWidgetItem*>::iterator p;
00613   //for(p=svIndex.begin(); p= svIndex.end(); p++){
00614     //delete p->second; // no need because they are destroyed already from statusView->clear();
00615   //}
00616   svIndex.clear();
00617 
00618   QTreeWidgetItem* svGeneralNode = new QTreeWidgetItem(statusView);
00619   svIndex.insert(pair<string, QTreeWidgetItem*>("general", svGeneralNode));
00620   svGeneralNode -> setText(0, "General");
00621   QTreeWidgetItem* svYearItem = new QTreeWidgetItem(svGeneralNode);
00622   svIndex.insert(pair<string, QTreeWidgetItem*>("general_year", svYearItem));
00623   svYearItem->setText(0, "year");
00624   svYearItem->setText(1, "0");
00625   QTreeWidgetItem* svTotalPlotsItem = new QTreeWidgetItem(svGeneralNode);
00626   svIndex.insert(pair<string, QTreeWidgetItem*>("general_total plots", svTotalPlotsItem));
00627   svTotalPlotsItem->setText(0, "total plots");
00628   svTotalPlotsItem->setText(1, "0");
00629   QTreeWidgetItem* svTotalLandItem = new QTreeWidgetItem(svGeneralNode);
00630   svIndex.insert(pair<string, QTreeWidgetItem*>("general_total land", svTotalLandItem));
00631   svTotalLandItem->setText(0, "total land");
00632   QTreeWidgetItem* svTotalAgrLandItem = new QTreeWidgetItem(svGeneralNode);
00633   svIndex.insert(pair<string, QTreeWidgetItem*>("general_total agr land", svTotalAgrLandItem));
00634   svTotalAgrLandItem->setText(0, "total agr land");
00635   QTreeWidgetItem* svOwnedAgrLandItem = new QTreeWidgetItem(svGeneralNode);
00636   svIndex.insert(pair<string, QTreeWidgetItem*>("general_owned agr land", svOwnedAgrLandItem));
00637   svOwnedAgrLandItem->setText(0, "owned agr land");
00638   QTreeWidgetItem* svRentedAgrLandItem = new QTreeWidgetItem(svGeneralNode);
00639   svIndex.insert(pair<string, QTreeWidgetItem*>("general_rented agr land", svRentedAgrLandItem));
00640   svRentedAgrLandItem->setText(0, "rented agr land");
00641 
00642   QTreeWidgetItem* svManagersNode = new QTreeWidgetItem(statusView);
00643   svIndex.insert(pair<string, QTreeWidgetItem*>("managers", svManagersNode));
00644   svManagersNode->setText(0,"Managers");
00645 
00646   QTreeWidgetItem* svAgentsNode = new QTreeWidgetItem(statusView);
00647   svIndex.insert(pair<string, QTreeWidgetItem*>("agents", svAgentsNode));
00648   svAgentsNode->setText(0,"Agents");
00649 
00650   // reset layer selector
00651   layerSelector->clear();
00652   // reset pixel info area
00653   pxInfoArea->setHtml("<i>Tip: Right click over a plot to retrieve its values across layers.</i>"
      );
00654   // reset log area
00655   logArea->clear();
00656   // reset map
00657 
00658   if (simulationCounter) logArea->append("***WARNING: You are running more simulations from the GUI
       without closing/reopening it. It should works, but there are no guarantees. The best way is to run only one
       simulation from the GUI, eventually closing and opening FFSM again for further simulations.");
00659   simulationCounter++;
00660 
00661 }
```

4.22.4.17 void resized ( ) [signal]

4.22.4.18 void resizeEvent (QResizeEvent * *event*) [protected]

Manage the event of resizing the application.

Definition at line 193 of file MainWindow.cpp.

```
00193                                               {
00194   emit resized();
00195 }
```

4.22.4.19 bool save ( ) [private], [slot]

Definition at line 337 of file MainWindow.cpp.

```
00337                  {
00338   if (curLogFileName.isEmpty()) {
00339     return saveAs();
00340   } else {
00341     cerr <<(curLogFileName.toStdString())<<endl;
00342     cerr <<(outputDirName.toStdString())<<endl;
00343     return saveLogFile(curLogFileName);
00344   }
00345   unsavedStatus = false;
00346   return true;
00347 }
```

4.22.4.20 bool saveAs ( ) [private], [slot]

Definition at line 350 of file MainWindow.cpp.

```
00350                    {
00351   QString logFileName = QFileDialog::getSaveFileName(
00352     this,
00353     tr("Save output log"),
00354     outputDirName,
00355     tr("Log files (*.log)\n" "All files (*.*)")
00356   );
00357   if (logFileName.isEmpty())
00358     return false;
00359   return saveLogFile(logFileName);
00360   unsavedStatus = false;
00361   return true;
00362 }
```

4.22.4.21 bool saveLogFile (const QString & *logFileName*)

Definition at line 365 of file MainWindow.cpp.

```
00365                                                   {
00366   QFile file(logFileName);
00367   if (!file.open(QIODevice::WriteOnly)) {
00368   QMessageBox::warning(this, tr("FFSM"),
00369      tr("Cannot write log file file %1:\n%2.")
00370     .arg(file.fileName())
00371     .arg(file.errorString()));
00372     return false;
00373   }
00374   //QString logAreaContent = logArea->toHtml();
00375   QString logAreaContent = logArea->toPlainText(); // Also available "toHtml()"
00376   QTextStream stream( &file );
00377   stream << logAreaContent;
00378   file.close();
00379 
00380   setCurrentLogFileName(logFileName);
00381   statusBar()->showMessage(tr("Log file saved"), 2000);
00382   unsavedStatus = false;
00383   return true;
00384 }
```

4.22.4.22 void selectedScenarioName (const QString & *scenarioName_h*) [signal]

4.22.4.23 void setCurrentLogFileName (const QString & *fileName*)

Definition at line 201 of file MainWindow.cpp.

```
00201                                                          {
00202   curLogFileName = fileName;
00203 }
```

4.22.4.24 void setCurrentModelFileName (const QString & *fileName*)

Definition at line 206 of file MainWindow.cpp.

```
00206                                                             {
00207   curModelFileName = fileName;
00208   //setWindowModified(false);
00209   modelMainThread.setInputFileName(
      curModelFileName);
00210
00211   QString shownName = "Untitled";
00212   if (!curModelFileName.isEmpty()) {
00213     shownName = strippedName(curModelFileName);
00214     recentFiles.removeAll(curModelFileName);
00215     recentFiles.prepend(curModelFileName);
00216     updateRecentFileActions();
00217   }
00218   setWindowTitle(tr("%2 - [%1]").arg(shownName).arg(tr("FFSM - Forest Sector Simulator")));
00219 }
```

4.22.4.25 void setModelFileName (const QString *curModelFileName_h*) `[inline]`

Definition at line 62 of file MainWindow.h.

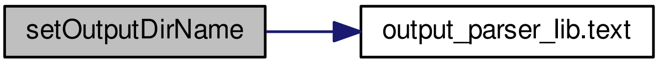
```
00062 {curModelFileName=curModelFileName_h;};
```

4.22.4.26 void setOutputDirName (string *outputDirName_h*) `[inline]`

Definition at line 66 of file MainWindow.h.

```
00066 {outputDirName = outputDirName_h.c_str();};
```

Here is the call graph for this function:



4.22.4.27 void setUnsavedStatus (bool *unsavedStatus_h*) `[inline]`, `[slot]`

Definition at line 65 of file MainWindow.h.

```
00065 {unsavedStatus = unsavedStatus_h;};
```

4.22.4.28 void showDocumentation ( ) [private], [slot]

Definition at line 585 of file MainWindow.cpp.

```
00585                                {
00586   QMessageBox::question(this, tr("FFSM Documentation"), // QMessageBox::information or
       QMessageBox::question
00587     tr("<h2>FFSM Documentation</h2>"
00588        "<p align=\"justify\">FFSM documentation is organised in three main categories: "
00589        "<p align=\"left\">(1) <b>official documentation</b> "
00590        "(comprising the <i>User Manual</i> and the <i>Reference Manual</i>); <br>(2) <b>contributed "
00591        "documentation</b> (<i>wiki</i>);<br>(3) <b>community project</b> (<i>forum</i> and <i>mailing
       list</i>). "
00592        "<p align=\"justify\">The documentation is located at "
00593        "<a href=\"http://www.ffsm-project.org/doc\">http://www.ffsm-project.org/doc</a>"
00594        "<p align=\"justify\">If you have chosen to instal a local copy of the documentation, "
00595        "you can access it also from the <i>Start menu</i>-><i>Programs</i>-><i>FFSM</i> "
00596        "(MS Windows) or directly from the following links (Linux):"
00597        "<br><a href=\"doc/userManual/regmasUserManual.pdf\">User Manual</a> "
00598        "&nbsp;-&nbsp;<a href=\"doc/referenceManual/html/index.html\">Reference Manual</a> "
00599        "<p>Tips:"
00600        "<br> - right click on a pixel to get its value across the layers;"
00601        "<br> - use the mouse and its wheel over the map to zoom/scroll it;"
00602        "</p>"
00603     ));
00604 }
```

4.22.4.29 void startModelMainThread ( ) [private], [slot]

Definition at line 394 of file MainWindow.cpp.

```
00394                                    {
00395   if (modelMainThread.isRunning()) {
00396     return ;
00397     cout <<"It seems that the model is already running..."<<endl;
00398     } else {
00399     logArea->clear();
00400     modelMainThread.start();
00401     unsavedStatus=true;
00402     }
00403 }
```

4.22.4.30 void stopModelMainThread ( ) [private], [slot]

Definition at line 406 of file MainWindow.cpp.

```
00406                                   {
00407   if (! modelMainThread.isRunning()) {
00408     return ;
00409   } else {
00410     modelMainThread.stop();
00411     modelMainThread.wait();
00412   }
00413 }
```

4.22.4.31 QString strippedName (const QString & *fullFileName*)

Definition at line 222 of file MainWindow.cpp.

```
00222                                                        {
00223   return QFileInfo(fullFileName).fileName();
00224 }
```

4.22.4.32 void switchToLayer (QString *layerName_h*)

Perform all the operation needed when switching layer:

- call mapBox to switch its current layer;
- call layerLegend to switch its layer); I don't think it is used anywhere, but any how.. it is here...

Definition at line 457 of file MainWindow.cpp.

```
00457                                              {
00458   mapBox->switchToLayer(layerName_h);
00459   int index = mapBox->getLayerIndex(layerName_h);
00460   layerSelector->setCurrentIndex(index);
00461   update();
00462 }
```

4.22.4.33 void switchToLayerFromLayerSelector (int *layerIndex_h*)

Definition at line 465 of file MainWindow.cpp.

```
00465                                                           {
00466   QString layerName= layerSelector->itemData(layerIndex_h, Qt::UserRole ).toString();
00467   mapBox->switchToLayer(layerName);
00468   update();
00469 }
```

4.22.4.34 void treeViewerAddItem (string *text,* string *itemID,* string *parentID*)

e.g. manager_farmer_manager agents or agent_12345_ownedHa

Definition at line 528 of file MainWindow.cpp.

```
00528                                                                         {
00529   // searching for the parent item...
00530   map<string, QTreeWidgetItem*>::iterator p;
00531   QTreeWidgetItem *parentItem;
00532
00533   p=svIndex.find(parentID);
00534   if(p != svIndex.end()){
00535     parentItem = p->second;
00536     QTreeWidgetItem *node = new QTreeWidgetItem(parentItem);
00537     svIndex.insert(pair<string, QTreeWidgetItem*>(itemID, node));
00538     node->setText(0, text.c_str());
00539   }
00540   else {
00541     QString tempString;
00542     QString tempString2 = itemID.c_str();
00543     QString tempString3 = parentID.c_str();
00544     tempString = "**** ERROR, Coud not add sub item "+tempString2+" to the Model Status Viewer. Parent item
       ("+tempString3+") doesn't found.";
00545     logArea->append(tempString);
00546   }
00547
00548 }
```

4.22.4.35 void treeViewerItemChangeValue (string *itemID,* string *newValue*)

Change value to an existing item in the Status Viewer.

Definition at line 485 of file MainWindow.cpp.

```
00485                                                                    {
00486
00487   map<string, QTreeWidgetItem*>::iterator p;
00488   p=svIndex.find(itemID);
00489   if(p != svIndex.end())
00490     p->second->setText(1,newValue.c_str());
00491   else {
00492     QString tempString;
00493     QString tempString2 = itemID.c_str();
00494     tempString = "**** ERROR, Coud not change value for item "+tempString2+" in the Model Status Viewer.
       Item doesn't found.";
00495     logArea->append(tempString);
00496   }
00497   return;
00498
00499 }
```

4.22.4.36 void treeViewerItemRemove (string *itemID*)

Definition at line 502 of file MainWindow.cpp.

```
00502                                                  {
00503   map<string, QTreeWidgetItem*>::iterator p;
00504   p=svIndex.find(itemID);
00505   if(p != svIndex.end()){
00506     QTreeWidgetItem *parent = p->second->parent();
00507     int index;
00508     if (parent) {
00509       index = parent->indexOfChild(p->second); //DONE: check if it works !!! While it should not ???? After
       15 years of simulation agents should be deleted, but htey are still here in the tree.. mayme it is true it
       is NOT working!!! To be checken. 20071108: It works, it works.. agents are deleted when go out of the model
00510       delete parent->takeChild(index);
00511       svIndex.erase(p);
00512     } else {
00513       QString tempString = "**** ERROR, I will not delete a top level item in the Model Satus Viewer";
00514       logArea->append(tempString);
00515     }
00516
00517   }
00518   else {
00519     QString tempString;
00520     QString tempString2 = itemID.c_str();
00521     tempString = "**** ERROR, Coud not delete for item "+tempString2+" in the Model Status Viewer. Item
       doesn't found.";
00522     //logArea->append(tempString); //20080111 lots of this errors when re-starting a simulation, so hidding
       them
00523   }
00524   return;
00525 }
```

4.22.4.37 void updateImage (QString *layerName_h,* const QImage & *image_h*)

Definition at line 478 of file MainWindow.cpp.

```
00478                                                                  {
00479   mapBox->updateImage(layerName_h, image_h);
00480   update();
00481 }
```

4.22.4.38 void updatePixel (QString *layerName_h*, int *x_h*, int *y_h*, QColor *color_h*)

Definition at line 472 of file MainWindow.cpp.

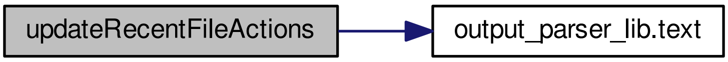
```
00472                                                                           {
00473   mapBox->updatePixel(layerName_h,x_h,y_h,color_h.rgb());
00474   update();
00475 }
```

4.22.4.39 void updateRecentFileActions ( ) [private]

Definition at line 227 of file MainWindow.cpp.

```
00227                                           {
00228   QMutableStringListIterator i(recentFiles);
00229   while (i.hasNext()) {
00230     if (!QFile::exists(i.next()))
00231       i.remove();
00232   }
00233 
00234   for (int j = 0; j < MaxRecentFiles; ++j) {
00235     if (j < recentFiles.count()) {
00236       QString text = tr("&%1 %2")
00237             .arg(j + 1)
00238             .arg(strippedName(recentFiles.at(j)));
00239       //cerr <<text.toStdString()<<endl;
00240       recentFileActions[j]->setText(text);
00241       recentFileActions[j]->setData(recentFiles.at(j));
00242       recentFileActions[j]->setVisible(true);
00243     } else {
00244       recentFileActions[j]->setVisible(false);
00245     }
00246   }
00247   separatorAction->setVisible(!recentFiles.isEmpty());
00248 }
```

Here is the call graph for this function:



4.22.4.40 void writeSettings ( ) [private]

Definition at line 386 of file MainWindow.cpp.

```
00386                                    {
00387   QSettings settings("LEF", "FFSM");
00388     settings.setValue("recentFiles", recentFiles);
00389 }
```

4.22.5 Member Data Documentation

4.22.5.1 QString curBaseDirectory [private]

Definition at line 114 of file MainWindow.h.

4.22.5.2 QString curLogFileName [private]

Definition at line 112 of file MainWindow.h.

4.22.5.3 QString curModelFileName [private]

Definition at line 113 of file MainWindow.h.

4.22.5.4 bool debugMsgsEnable [private]

Allow debug messages to be show in the logArea.

Definition at line 119 of file MainWindow.h.

4.22.5.5 QLabel∗ mainSBLabel [private]

Status bar main label.

Definition at line 109 of file MainWindow.h.

4.22.5.6 ThreadManager modelMainThread [private]

Definition at line 107 of file MainWindow.h.

4.22.5.7 QString outputDirName [private]

Definition at line 111 of file MainWindow.h.

4.22.5.8 QAction∗ recentFileActions[MaxRecentFiles] [private]

Definition at line 117 of file MainWindow.h.

4.22.5.9 QStringList recentFiles [private]

Definition at line 115 of file MainWindow.h.

4.22.5.10 ScenarioSelectionWidget∗ scenarioWidget [private]

Definition at line 120 of file MainWindow.h.

4.22.5.11 QAction∗ separatorAction [private]

Definition at line 118 of file MainWindow.h.

4.22.5.12 map<string, QTreeWidgetItem∗> svIndex [private]

Map containing the ID and the pointers to the status viewer.

Ids are based on the name of the item:

- general
- general_{name}
- managers
- manager_{managerID}
- manager_{managerID}_{name}
- agents
- agent_{agentUniqueID}
- agent_{agentUniqueID}_{name}

Definition at line 132 of file MainWindow.h.

4.22.5.13 bool unsavedStatus [private]

Definition at line 110 of file MainWindow.h.

4.22.5.14 QLabel∗ yearSBLabel [private]

Status bar current year label.

Definition at line 108 of file MainWindow.h.

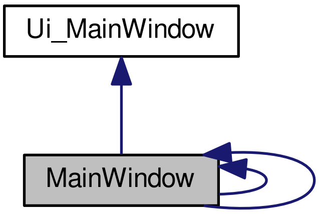
The documentation for this class was generated from the following files:

- /home/lobianco/git/ffsm_pp/src/MainWindow.h
- /home/lobianco/git/ffsm_pp/src/MainWindow.cpp

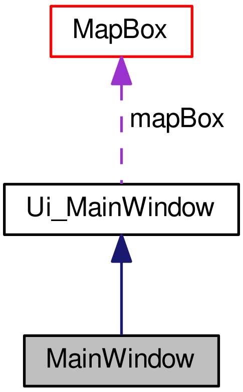
4.23 MainWindow Class Reference

`#include <ui_MainWindow.h>`

Inheritance diagram for MainWindow:



Collaboration diagram for MainWindow:



Additional Inherited Members

4.23.1 Detailed Description

Definition at line 337 of file ui_MainWindow.h.

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

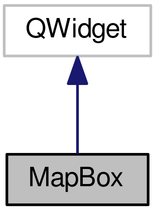
- /home/lobianco/git/ffsm_pp/src/ui_MainWindow.h

4.24 MapBox Class Reference

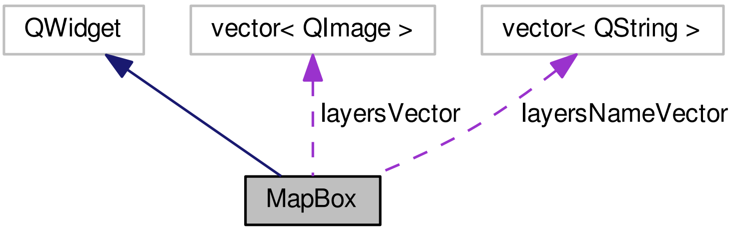
Widget to display the maps of various spacial aspects of the model.

`#include <MapBox.h>`

Inheritance diagram for MapBox:



Collaboration diagram for MapBox:



Public Slots

- void updatePixel (QString layerName_h, int x_h, int y_h, QColor color_h)
- void updateImage (QString layerName_h, const QImage &image_h)
- void switchToLayer (QString layerName_h)

  Change the layer that currentLayer and currentLayerName points.
- void addLayer (QString layerName_h)
- void fitInWindow ()
- void zoom (double zoomFactor)
- void scroll (int deltaX, int deltaY)

Signals

- void queryRequestOnPx (int px_ID, int currentLayerIndex, bool newRequest)

Public Member Functions

- MapBox (QWidget ∗parent=0)
- int getLayerIndex (QString layerName_h="")

  Return the index of the specified layer (null to ask for the current one)

Private Member Functions

- void updatePixmap (const QImage &image, bool reFit=false)
- void paintEvent (QPaintEvent ∗event)

  Reimplementation of the standard paintEvent method.
- void prepareQueryEvent (QPoint click)
- void keyPressEvent (QKeyEvent ∗event)
- void wheelEvent (QWheelEvent ∗event)
- void mousePressEvent (QMouseEvent ∗event)
- void mouseMoveEvent (QMouseEvent ∗event)

Private Attributes

- vector< QImage > layersVector

  Vector of QImages.
- vector< QString > layersNameVector

  Vector of layer names.
- QImage currentLayer
- QString currentLayerName
- QPoint lastDragPos
- double sx1
- double sy1
- double sx2
- double sy2

  coordinates of corner pixels of source - pixmap - rectangle
- double dx1
- double dy1
- double dx2
- double dy2

  coordinates of corner pixels of destination - widget - rectangle

4.24.1 Detailed Description

Widget to display the maps of various spacial aspects of the model.

This class is based on QImage. It pick-ups from layersVector the choosed layer and display it.
It has methods to change the individual pixels or the whole image of a layer.

Definition at line 41 of file MapBox.h.

4.24.2 Constructor & Destructor Documentation

4.24.2.1 MapBox (QWidget ∗ *parent* = 0)

Definition at line 35 of file MapBox.cpp.

```
00035                                  :QWidget(parent) {
00036 
00037   currentLayerName = "";
00038   setCursor(Qt::CrossCursor);
00039 
00040   // setting source and destination init corners..
00041   sx1 = 0;
00042   sy1 = 0;
00043   sx2 = this->width();
00044   sy2 = this->height();
00045   dx1 = 0;
00046   dy1 = 0;
00047   dx2 = this->width();
00048   dy2 = this->height();
00049 }
```

4.24.3 Member Function Documentation

4.24.3.1 void addLayer (QString *layerName_h*) [slot]

Definition at line 135 of file MapBox.cpp.

```
00135                                    {
00136   static int counter = 0;
00137   QImage newlayer = QImage(this->width(), this->height(), QImage::Format_RGB32);
00138   newlayer.fill(qRgb(255, 255, 255));
00139   layersVector.push_back(newlayer);
00140   layersNameVector.push_back(layerName_h);
00141   if (counter == 0) {
00142     currentLayerName = layerName_h;
00143     currentLayer = layersVector.at(0);
00144   }
00145   counter ++;
00146 }
```

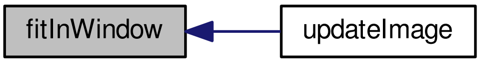
4.24.3.2 void fitInWindow ( ) [slot]

Definition at line 217 of file MapBox.cpp.

Referenced by updateImage().

```
00217                         {
00218
00219   QPixmap pixmap = QPixmap::fromImage(currentLayer);
00220   double tempXScale = ( (double) this->width()) / ((double)pixmap.width());
00221   double tempYScale = ( (double) this->height())/ ((double)pixmap.height());
00222
00223   sx1 = 0;
00224   sy1 = 0;
00225   sx2 = pixmap.width();
00226   sy2 = pixmap.height();
00227   dx1 = 0;
00228   dy1 = 0;
00229
00230   if ( tempXScale >= tempYScale){
00231     dx2 = ((double)pixmap.width())*tempYScale;
00232     dy2 = this->height();
00233   } else {
00234     dx2 = this->width();
00235     dy2 = ((double)pixmap.height())*tempXScale;
00236   }
00237   update();
00238 }
```

Here is the caller graph for this function:



4.24.3.3 int getLayerIndex (QString *layerName_h* = "")

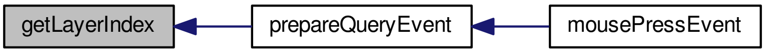
Return the index of the specified layer (null to ask for the current one)

Definition at line 123 of file MapBox.cpp.

Referenced by prepareQueryEvent().

```
00123                                         {
00124   if( layerName_h == "") layerName_h = currentLayerName;
00125   for (uint i=0;i<layersVector.size();i++){
00126     if (layersNameVector.at(i) == layerName_h){
00127       return i;
00128     }
00129   }
00130   cout << "*** ERROR in MapBox:getLayerIndex(): layerName_h "<< qPrintable(layerName_h) << " not found "<<
      endl;
00131   return -1;
00132 }
```

Here is the caller graph for this function:

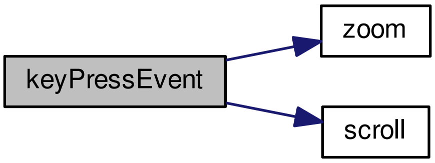


4.24.3.4 void keyPressEvent (QKeyEvent ∗ *event*) [private]

Definition at line 149 of file MapBox.cpp.

```
00149                                       {
00150   switch (event->key()) {
00151     case Qt::Key_Plus:
00152     zoom(ZoomInFactor);
00153     break;
00154   case Qt::Key_Minus:
00155     zoom(ZoomOutFactor);
00156     break;
00157   case Qt::Key_Left:
00158     scroll(+ScrollStep, 0);
00159     break;
00160   case Qt::Key_Right:
00161     scroll(-ScrollStep, 0);
00162     break;
00163   case Qt::Key_Down:
00164     scroll(0, -ScrollStep);
00165     break;
00166   case Qt::Key_Up:
00167     scroll(0, +ScrollStep);
00168     break;
00169   default:
00170     QWidget::keyPressEvent(event);
00171   }
00172 }
```

Here is the call graph for this function:

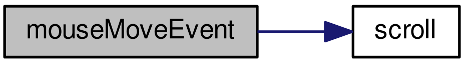


4.24.3.5 void mouseMoveEvent (QMouseEvent * *event*) [private]

Definition at line 209 of file MapBox.cpp.

```
00209                                              {
00210   if (event->buttons() & Qt::LeftButton) {
00211     scroll(event->pos().x()-lastDragPos.x(), event->pos().y()-
      lastDragPos.y());
00212     lastDragPos = event->pos();
00213     update();
00214   }
00215 }
```

Here is the call graph for this function:

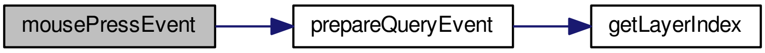


4.24.3.6 void mousePressEvent (QMouseEvent * *event*) [private]

Definition at line 182 of file MapBox.cpp.

```
00182                                              {
00183   if (event->button() == Qt::LeftButton){
00184     lastDragPos = event->pos();
00185   }
00186   else if (event->button() == Qt::RightButton){
00187     prepareQueryEvent(event->pos());
00188   }
00189 }
```

Here is the call graph for this function:



4.24.3.7 void paintEvent (QPaintEvent * *event*) `[private]`

Reimplementation of the standard paintEvent method.

We paint the image pixel by pixel picking up the colors from the map pointed by currentLayer.

Definition at line 55 of file MapBox.cpp.

```
00055                                       {
00056
00057   if (layersVector.size() < 1) return;
00058   QPainter painter(this);
00059   painter.fillRect(rect(), Qt::lightGray  );
00060   QPixmap pixmap = QPixmap::fromImage(currentLayer); // It doesn't get autmoatically refreshed
       if I use a separate function to update the pixmap from the image
00061   QRectF source     (sx1, sy1, sx2-sx1, sy2-sy1); // the second point is in coordinates
       origin of the firt point !!!!
00062   QRectF destination(dx1, dy1, dx2-dx1, dy2-dy1); // the second point is in coordinates
       origin of the firt point !!!!
00063   /*
00064   This is the main function of the widget... the good pointa are:
00065   A) It takes into account the low level details of scaling, such interpolation
00066   B) If the destination is outside the widgets bounds, it doesn't matter. It make its job on the widget
       without any error (in this sence it isnot like an array luckily...)
00067   */
00068   painter.drawPixmap(destination, pixmap, source);
00069
00070 }
```

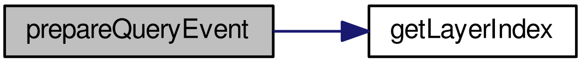
4.24.3.8 void prepareQueryEvent (QPoint *click*) `[private]`

Definition at line 192 of file MapBox.cpp.

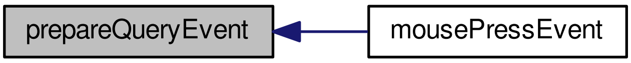
Referenced by mousePressEvent().

```
00192                                       {
00193   double cx = ((double) click.x()); //clicked x, casted to double
00194   double cy = ((double) click.y()); //clicked y, casted to double
00195   int    mx, my = 0; // outputed x and y
00196   int    px_ID;  // pixel ID
00197   int    layerIndex = getLayerIndex();
00198   // checking it is not out of the destination border range..
00199   if (cx>dx2 || cx<dx1 || cy>dy2 || cy<dy1) return;
00200   mx = ( (int) (cx-dx1) * (sx2-sx1)/(dx2-dx1) + sx1); // casting to int, not round() !!
00201   my = ( (int) (cy-dy1) * (sy2-sy1)/(dy2-dy1) + sy1); // casting to int, not round() !!
00202   px_ID = mx+my*(sx2-sx1);
00203   emit queryRequestOnPx(px_ID, layerIndex, true);
00204
00205 }
```

Here is the call graph for this function:



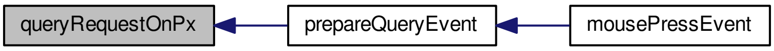
Here is the caller graph for this function:



4.24.3.9 void queryRequestOnPx (int *px_ID,* int *currentLayerIndex,* bool *newRequest*) [signal]

Referenced by prepareQueryEvent().

Here is the caller graph for this function:



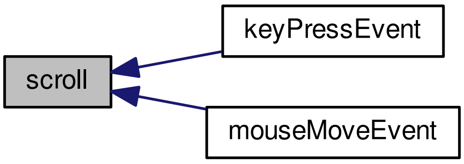
4.24.3.10 void scroll (int *deltaX,* int *deltaY*) [slot]

Definition at line 255 of file MapBox.cpp.

Referenced by keyPressEvent(), and mouseMoveEvent().

```
00255                                      {
00256   dx1 += ((double) deltaX);
00257   dx2 += ((double) deltaX);
00258   dy1 += ((double) deltaY);
00259   dy2 += ((double) deltaY);
00260   update();
00261 }
```

Here is the caller graph for this function:



4.24.3.11 void switchToLayer (QString *layerName_h*) [slot]

Change the layer that currentLayer and currentLayerName points.

Definition at line 108 of file MapBox.cpp.

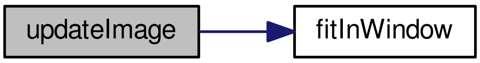
```
00108                                          {
00109   if (layerName_h != currentLayerName){
00110     for (uint i=0;i<layersVector.size();i++){
00111       if (layersNameVector.at(i) == layerName_h){
00112         currentLayer = layersVector.at(i);
00113         currentLayerName = layerName_h;
00114         update();
00115         return;
00116       }
00117     }
00118     cout << "*** ERROR in MapBox::switchToLayer(): layerName_h "<< qPrintable(layerName_h) << " not found "
      << endl;
00119   }
00120 }
```

4.24.3.12 void updateImage (QString *layerName_h*, const QImage & *image_h*) [slot]

Definition at line 87 of file MapBox.cpp.

```
00087                                                                {
00088   static int counter = 0;
00089   for (uint i=0;i<layersVector.size();i++){
00090     if (layersNameVector.at(i) == layerName_h){
00091       layersVector.at(i) = image_h;
00092       if(layerName_h == currentLayerName){
00093         currentLayer = layersVector.at(i);
00094         update();
00095       }
00096       if (counter == 0) { // that's the first image we got !!
00097         fitInWindow();
00098       }
00099       counter ++;
00100       return;
00101     }
00102   }
00103   counter ++;
00104   cout << "*** ERROR in MapBox::updateImage(): layerName_h "<< qPrintable(layerName_h) << " not found "<<
      endl;
00105 }
```

Here is the call graph for this function:



4.24.3.13 void updatePixel (QString *layerName_h*, int *x_h*, int *y_h*, QColor *color_h*) [slot]

Definition at line 73 of file MapBox.cpp.

```
00073                                                                           {
00074   for (uint i=0;i<layersVector.size();i++){
00075     if (layersNameVector.at(i) == layerName_h){
00076       layersVector.at(i).setPixel(x_h, y_h, color_h.rgb());
00077       if(layerName_h == currentLayerName){
00078         currentLayer = layersVector.at(i);
00079         update();
00080       }
00081       return;
00082     }
00083   }
00084 }
```

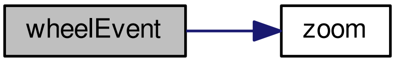
4.24.3.14 void updatePixmap (const QImage & *image,* bool *reFit* = `false`) `[private]`

4.24.3.15 void wheelEvent (QWheelEvent ∗ *event*) `[private]`

Definition at line 175 of file MapBox.cpp.

```
00175                                 {
00176   int numDegrees = event->delta() / 8;
00177   double numSteps = numDegrees / 15.0f;
00178   zoom(pow(ZoomInFactor, numSteps));
00179 }
```

Here is the call graph for this function:



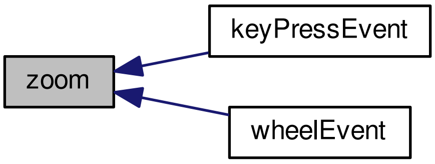
4.24.3.16 void zoom (double *zoomFactor*) `[slot]`

Definition at line 241 of file MapBox.cpp.

Referenced by keyPressEvent(), and wheelEvent().

```
00241                            {
00242   double dx1new, dx2new, dy1new, dy2new;
00243   dx1new = dx2- (dx2-dx1)* ( 1+ (zoomFactor-1)/2 );
00244   dx2new = dx1+ (dx2-dx1)* ( 1+ (zoomFactor-1)/2 );
00245   dy1new = dy2- (dy2-dy1)* ( 1+ (zoomFactor-1)/2 );
00246   dy2new = dy1+ (dy2-dy1)* ( 1+ (zoomFactor-1)/2 );
00247   dx1 = dx1new;
00248   dy1 = dy1new;
00249   dx2 = dx2new;
00250   dy2 = dy2new;
00251   update();
00252 }
```

Here is the caller graph for this function:



4.24.4 Member Data Documentation

4.24.4.1 QImage currentLayer `[private]`

Definition at line 70 of file MapBox.h.

Referenced by addLayer(), fitInWindow(), paintEvent(), switchToLayer(), updateImage(), and updatePixel().

4.24.4.2 QString currentLayerName `[private]`

Definition at line 71 of file MapBox.h.

Referenced by addLayer(), getLayerIndex(), MapBox(), switchToLayer(), updateImage(), and updatePixel().

4.24.4.3 double dx1 `[private]`

Definition at line 74 of file MapBox.h.

Referenced by fitInWindow(), MapBox(), paintEvent(), prepareQueryEvent(), scroll(), and zoom().

4.24.4.4 double dx2 `[private]`

Definition at line 74 of file MapBox.h.

Referenced by fitInWindow(), MapBox(), paintEvent(), prepareQueryEvent(), scroll(), and zoom().

4.24.4.5 double dy1 `[private]`

Definition at line 74 of file MapBox.h.

Referenced by fitInWindow(), MapBox(), paintEvent(), prepareQueryEvent(), scroll(), and zoom().

4.24.4.6 double dy2 `[private]`

coordinates of corner pixels of destination - widget - rectangle

Definition at line 74 of file MapBox.h.

Referenced by fitInWindow(), MapBox(), paintEvent(), prepareQueryEvent(), scroll(), and zoom().

4.24.4.7 QPoint lastDragPos `[private]`

Definition at line 72 of file MapBox.h.

Referenced by mouseMoveEvent(), and mousePressEvent().

4.24.4.8 vector<QString> layersNameVector `[private]`

Vector of layer names.

Definition at line 69 of file MapBox.h.

Referenced by addLayer(), getLayerIndex(), switchToLayer(), updateImage(), and updatePixel().

4.24.4.9 vector< QImage > layersVector [private]

Vector of QImages.

Definition at line 68 of file MapBox.h.

Referenced by addLayer(), getLayerIndex(), paintEvent(), switchToLayer(), updateImage(), and updatePixel().

4.24.4.10 double sx1 [private]

Definition at line 73 of file MapBox.h.

Referenced by fitInWindow(), MapBox(), paintEvent(), and prepareQueryEvent().

4.24.4.11 double sx2 [private]

Definition at line 73 of file MapBox.h.

Referenced by fitInWindow(), MapBox(), paintEvent(), and prepareQueryEvent().

4.24.4.12 double sy1 [private]

Definition at line 73 of file MapBox.h.

Referenced by fitInWindow(), MapBox(), paintEvent(), and prepareQueryEvent().

4.24.4.13 double sy2 [private]

coordinates of corner pixels of source - pixmap - rectangle

Definition at line 73 of file MapBox.h.

Referenced by fitInWindow(), MapBox(), paintEvent(), and prepareQueryEvent().

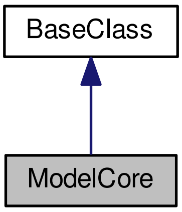
The documentation for this class was generated from the following files:

- /home/lobianco/git/ffsm_pp/src/MapBox.h
- /home/lobianco/git/ffsm_pp/src/MapBox.cpp

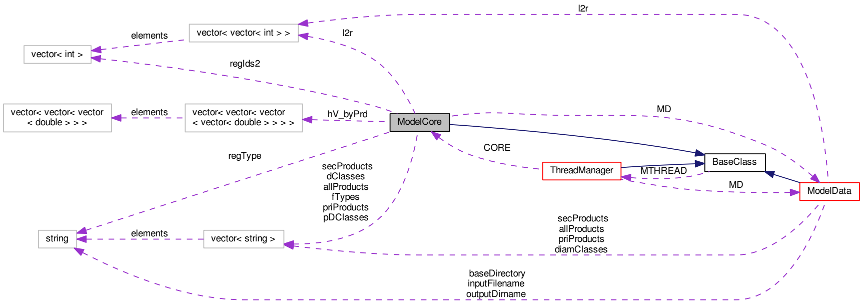
4.25 ModelCore Class Reference

`#include <ModelCore.h>`

Inheritance diagram for ModelCore:



Collaboration diagram for ModelCore:



Public Member Functions

- ModelCore (ThreadManager ∗MTHREAD_h)
- ∼ModelCore ()
- void runInitPeriod ()
- void runSimulationYear ()
- void initMarketModule ()

  computes st and pw for second year and several needed-only-at-t0-vars for the market module
- void runMarketModule ()

  computes st (supply total) and pw (weighted price). Optimisation inside.
- void runBiologicalModule ()

  computes hV, hArea and new vol at end of year
- void runManagementModule ()

  computes regArea and expectedReturns
- void cacheSettings ()

  just cache exogenous settings from ModelData
- void cachePixelExogenousData ()

computes pixel level tp, meta and mort
- void computeInventary ()

  in=f(vol_t-1)
- void computeCumulativeData ()

  computes cumTp, vHa, cumTp_exp, vHa_exp,
- void updateMapAreas ()

  computes forArea_{ft}

Private Member Functions

- double gpd (const string &type_h, const int &regId_h, const string &prodId_h, const int &year=DATA_NOW, const string &freeDim_h="") const
- double gfd (const string &type_h, const int &regId_h, const string &forType_h, const string &freeDim_h, const int &year=DATA_NOW) const
- void spd (const double &value_h, const string &type_h, const int &regId_h, const string &prodId_h, const int &year=DATA_NOW, const bool &allowCreate=false, const string &freeDim_h="") const
- void sfd (const double &value_h, const string &type_h, const int &regId_h, const string &forType_h, const string &freeDim_h, const int &year=DATA_NOW, const bool &allowCreate=false) const
- bool app (const string &prod_h, const string &forType_h, const string &dClass_h) const

Private Attributes

- ModelData $*$ MD
- int firstYear
- int secondYear
- int thirdYear
- int WL2
- vector< int > regIds2
- vector< string > priProducts
- vector< string > secProducts
- vector< string > allProducts
- vector< string > dClasses
- vector< string > pDClasses
- vector< string > fTypes
- vector< vector< int > > l2r
- string regType
- double expType
- double mr
- vector< vector< vector< vector< double > > > > hV_byPrd
- bool rescaleFrequencies

Additional Inherited Members

4.25.1 Detailed Description

Definition at line 43 of file ModelCore.h.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 ModelCore (ThreadManager ∗ *MTHREAD_h*)

Definition at line 37 of file ModelCore.cpp.

```
00037                                            {
00038   MTHREAD = MTHREAD_h;
00039 }
```

4.25.2.2 ∼ModelCore ( )

Definition at line 41 of file ModelCore.cpp.

```
00041                        {
00042
00043 }
```

4.25.3 Member Function Documentation

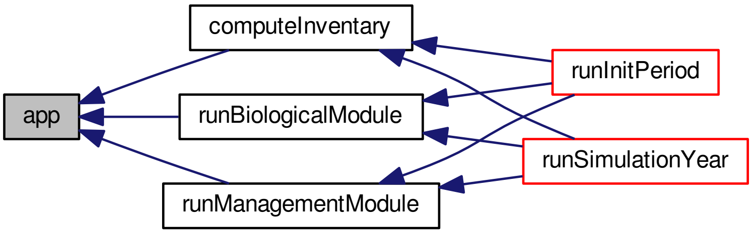
4.25.3.1 bool app (const string & *prod_h,* const string & *forType_h,* const string & *dClass_h*) const [inline], [private]

Definition at line 70 of file ModelCore.h.

Referenced by computeInventary(), runBiologicalModule(), and runManagementModule().

```
00070 {return MTHREAD->MD->assessProdPossibility(prod_h, forType_h, dClass_h);};
```

Here is the caller graph for this function:



4.25.3.2 void cachePixelExogenousData ( )

computes pixel level tp, meta and mort

4.25.3.3 void cacheSettings ( )

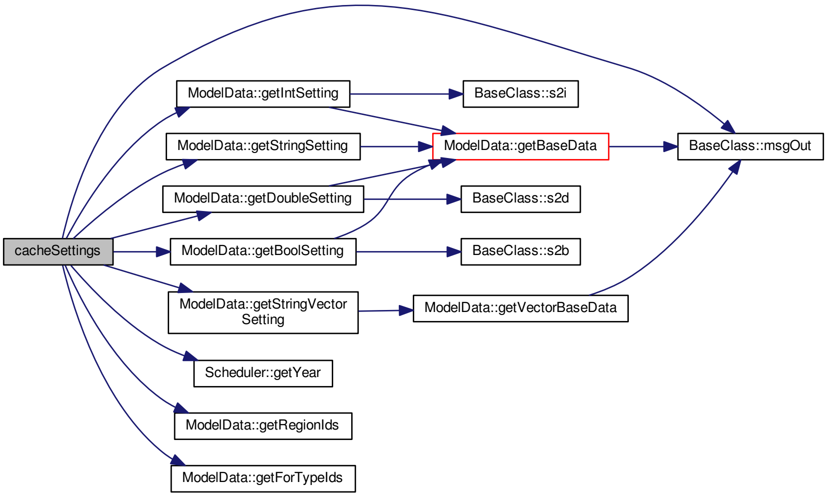
just cache exogenous settings from ModelData

Definition at line 692 of file ModelCore.cpp.

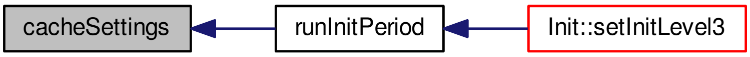
Referenced by runInitPeriod().

```
00692                         {
00693   msgOut(MSG_INFO, "Cashing initial model settings..");
00694   int currentYear = MTHREAD->SCD->getYear();
00695
00696   MD = MTHREAD->MD;
00697   firstYear = MD->getIntSetting("initialYear");
00698   secondYear = firstYear+1;
00699   thirdYear  = firstYear+2;
00700   WL2 = MD->getIntSetting("worldCodeLev2");
00701   regIds2 = MD->getRegionIds(2);
00702   priProducts = MD->getStringVectorSetting("priProducts");
00703   secProducts = MD->getStringVectorSetting("secProducts");
00704   allProducts = priProducts;
00705   allProducts.insert( allProducts.end(), secProducts.begin(),
      secProducts.end() );
00706   dClasses = MD->getStringVectorSetting("dClasses");
00707   pDClasses; // production diameter classes: exclude the fist diameter class below 15 cm
00708   pDClasses.insert(pDClasses.end(), dClasses.begin()+1,
      dClasses.end() );
00709   fTypes= MD->getForTypeIds();
00710   l2r = MD->getRegionIds();
00711   regType = MTHREAD->MD->getStringSetting("regType"); // how the
       regeneration should be computed (exogenous, from hr, from allocation choises)
00712   expType = MD->getDoubleSetting("expType");
00713   rescaleFrequencies = MD->getBoolSetting("rescaleFrequencies");
00714   if((expType<0 || expType>1) && expType != -1){
00715     msgOut(MSG_CRITICAL_ERROR, "expType parameter must be between 1 (expectations)
       and 0 (adaptative) or -1 (fixed).");
00716   }
00717   mr = MD->getDoubleSetting("mr");
00718 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.25.3.4 void computeCumulativeData ( )

computes cumTp, vHa, cumTp_exp, vHa_exp,

Computing some fully exogenous parameters that require complex operations, e.g. cumulative time of passage or volume per hectare. This happen at the very beginning of the init period and after each simulated year

It doesn't include tp and mort multipliers, but this could be added as now there is a regional versiopn of them and not just a pixel version.

```
param expType Specify how the forest owners (those that make the investments) behave will be the time of p
Will forest owners behave adaptively believing the time of passage between diameter classes will be like t
```

For compatibility with the GAMS code, a -1 value means using initial simulation tp values (fixed cumTp)."

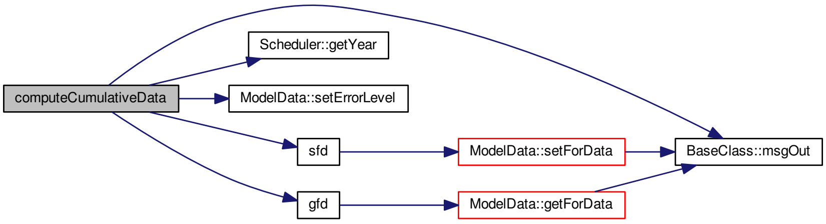
Definition at line 727 of file ModelCore.cpp.

Referenced by runInitPeriod(), and runSimulationYear().

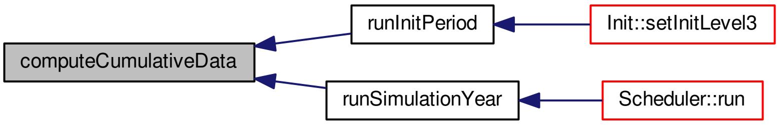
```
00727                                   {
00728
00729     msgOut(MSG_INFO, "Starting computing some cumulative values..");
00730     int thisYear      =  MTHREAD->SCD->getYear();
00731
00732     // debug
00733     //cout << "cumTp and vHa by dc:" << endl;
00734     //cout << "regId|ft|varName|0|15|25|35|45|55|65|75|85|95|150|" << endl;
00735
00736     for(uint r2= 0; r2<regIds2.size();r2++){
00737       int regId = regIds2[r2];
00738       for(uint j=0;j<fTypes.size();j++){
00739         string ft = fTypes[j];
00740         // calculating the cumulative time of passage and the (cumulativelly generated) vHa for each
       diameter class (depending on forest owners diam growth expectations)
00741         //loop(u$(ord(u)=1),
00742         //   cumTp(u,i,lambda,essence) = tp_u1(i,essence,lambda);
00743         //);
00744         //loop(u$(ord(u)>1),
00745         //   cumTp(u,i,lambda,essence) = cumTp(u-1,i,lambda,essence)+tp(u-1,i,lambda,essence);
00746         //);
00747         ////ceil(x) DNLP returns the smallest integer number greater than or equal to x
00748         //loop( (u,i,lambda,essence),
00749         //  cumTp(u,i,lambda,essence) =   ceil(cumTp(u,i,lambda,essence));
00750         //);
00751         /**
00752         param expType Specify how the forest owners (those that make the investments) behave will be the
       time of passage in the future in order to calculate the cumulative time of passage in turn used to discount
       future revenues.
00753         Will forest owners behave adaptively believing the time of passage between diameter classes will be
       like the observed one at time they make decision (0) or they will have full expectations believing
       forecasts (1) or something in the middle ?
00754 For compatibility with the GAMS code, a -1 value means using initial simulation tp values (fixed cumTp)."
00755         */
00756                 vector <double> cumTp_temp;        // cumulative time of passage to REACH a diameter class
       (tp is to LEAVE to the next one)
00757                 vector <double> vHa_temp;          // volume at hectar by each diameter class [m^3/ha]
00758                 vector <double> cumAlive_temp;     // cumulated alive rate to reach a given diameter class
```

```
                vector <double> cumTp_exp_temp;    // "expected" version of cumTp
                vector <double> vHa_exp_temp;      // "expected" version of vHa
                vector <double> cumAlive_exp_temp; // "expected" version of cumMort

        MD->setErrorLevel(MSG_NO_MSG); // as otherwise on 2007 otherwise sfd() will complain that is filling multiple years (2006 and 2007)
        for (uint u=0; u<dClasses.size(); u++){
          string dc = dClasses[u];
          double cumTp_u, cumTp_u_exp, cumTp_u_noExp, cumTp_u_fullExp;
          double vHa_u, vHa_u_exp, vHa_u_noExp, vHa_u_fullExp, beta, beta_exp, beta_noExp, beta_fullExp, mort, mort_exp, mort_noExp, mort_fullExp;
          double tp_u, tp_exp;
                    double cumAlive_u, cumAlive_exp_u;

          if(u==0) {
            // first diameter class.. expected  and real values are the same (0)
            cumTp_u = 0.;
            vHa_u   = 0.;
                        cumAlive_u = 1.;
            cumTp_temp.push_back(cumTp_u);
            cumTp_exp_temp.push_back(cumTp_u);
            vHa_temp.push_back(vHa_u);
            vHa_exp_temp.push_back(vHa_u);
                        cumAlive_temp.push_back(cumAlive_u);
                        cumAlive_exp_temp.push_back(cumAlive_u);
            sfd(cumTp_u,"cumTp",regId,ft,dc,DATA_NOW,true);
            sfd(cumTp_u,"cumTp_exp",regId,ft,dc,DATA_NOW,true);
            sfd(vHa_u,  "vHa",regId,ft,dc,DATA_NOW,true);
            sfd(vHa_u,  "vHa_exp",regId,ft,dc,DATA_NOW,true);
                        sfd(cumAlive_u,"cumAlive",regId,ft,dc,DATA_NOW,true);
                        sfd(cumAlive_u,"cumAlive_exp",regId,ft,dc,DATA_NOW,true);
          } else {
            // other diameter classes.. first dealing with real values and then with expected ones..
            // real values..
            cumTp_u = cumTp_temp[u-1] + gfd("tp",regId,ft,dClasses[u-1],thisYear); // it adds to the time of passage to reach the previous diameter class the time of passage that there should be to reach this diameter class in the year where the previous diameter class will be reached
            if (u==1){
              vHa_u = gfd("entryVolHa",regId,ft,"",thisYear);
                            mort = 0.; // not info about mortality first diameter class ("00")
            } else {
              mort = 1-pow(1-gfd("mortCoef",regId,ft,dClasses[u-1],thisYear), gfd("tp",regId,ft,dClasses[u-1],thisYear)); // mortality of the previous diameter class
                            beta = gfd("betaCoef",regId,ft,dc, thisYear);
              vHa_u = vHa_temp[u-1]*beta*(1-mort);
            }
                        cumAlive_u = max(0.,cumAlive_temp[u-1]*(1-mort));
                        cumAlive_temp.push_back(cumAlive_u);
            cumTp_temp.push_back(cumTp_u);
            vHa_temp.push_back(vHa_u);
            sfd(cumTp_u,"cumTp",regId,ft,dc,DATA_NOW,true);
            sfd(vHa_u,"vHa",regId,ft,dc,DATA_NOW,true);
                        sfd(cumAlive_u,"cumAlive",regId,ft,dc,DATA_NOW,true);

            // expected values..
            if (expType == -1){
              cumTp_u_exp = cumTp_exp_temp[u-1]+gfd("tp",regId,ft,dClasses[u-1], firstYear); // it adds to the time of passage to reach the previous diameter class the time of passage that there should be to reach this diameter class in the year where the previous diameter class will be reached
              cumTp_exp_temp.push_back(cumTp_u_exp);
              if(u==1) {
                vHa_u_exp = gfd("entryVolHa",regId,ft,"",firstYear);
                                mort_exp = 0.; // not info about mortality first diameter class ("00")
              } else {
                mort_exp   = 1-pow(1-gfd("mortCoef",regId,ft,dClasses[u-1], firstYear),gfd("tp",regId,ft,dClasses[u-1],firstYear)) ; // mortality rate of previous diameter class
                beta_exp   = gfd("betaCoef",regId,ft,dc, firstYear);
                vHa_u_exp = vHa_exp_temp[u-1]*beta_exp*(1-mort_exp);
              }
            } else {
              cumTp_u_noExp   = cumTp_exp_temp[u-1]+gfd("tp",regId,ft, dClasses[u-1]);
              cumTp_u_fullExp = cumTp_exp_temp[u-1]+gfd("tp",regId,ft, dClasses[u-1],thisYear+ceil(cumTp_exp_temp[u-1])); // it adds to the time of passage to reach the previous diameter class the time of passage that there should be to reach this diameter class in the year where the previous diameter class will be reached
              cumTp_u_exp     = cumTp_u_fullExp*expType+cumTp_u_noExp*(1-expType);
              cumTp_exp_temp.push_back(cumTp_u_exp);
              if(u==1) {
                vHa_u_noExp   = gfd("entryVolHa",regId,ft,"",DATA_NOW);
                vHa_u_fullExp = gfd("entryVolHa",regId,ft,"",thisYear+ceil(cumTp_u));
                vHa_u_exp     = vHa_u_fullExp*expType+vHa_u_noExp*(1-expType);
                                mort_exp      = 0. ;
              } else {
                mort_noExp   = 1-pow(1-gfd("mortCoef",regId,ft,dClasses[u-1],DATA_NOW),cumTp_exp_temp[u]-cumTp_exp_temp[u-1]);
                mort_fullExp = 1-pow(1-gfd("mortCoef",regId,ft,dClasses[u-1],thisYear+ceil(cumTp_temp[u-1])),cumTp_exp_temp[u]-cumTp_exp_temp[u-1]); // mortality of the previous diameter class
                beta_noExp   = gfd("betaCoef",regId,ft,dc, DATA_NOW);
                beta_fullExp = gfd("betaCoef",regId,ft,dc, thisYear+ceil(cumTp_u));
                mort_exp     = mort_fullExp*expType+mort_noExp*(1-expType);
                beta_exp     = beta_fullExp*expType+beta_noExp*(1-expType);
                vHa_u_exp    = vHa_exp_temp[u-1]*beta_exp*(1-mort_exp);
              }
            }
            vHa_exp_temp.push_back(vHa_u_exp);
                        cumAlive_exp_u = max(0.,cumAlive_exp_temp[u-1]*(1-mort_exp));
                        cumAlive_exp_temp.push_back(cumAlive_exp_u);
            sfd(cumTp_u_exp,"cumTp_exp",regId,ft,dc,DATA_NOW,true);
            sfd(vHa_u_exp,  "vHa_exp",regId,ft,dc,DATA_NOW,true);
                        sfd(cumAlive_exp_u,"cumAlive_exp",regId,ft,dc,DATA_NOW,true);
                        //sfd(cumMort_u_exp,  "cumMort_exp",regId,ft,dc,DATA_NOW,true);

                        //cout << "********" << endl;
                        //cout << "dc: " << dClasses[u] << endl ;
                        //cout << "mort: " << mort << endl;
                        //cout << "mort_exp: " << mort_exp << endl;
                        //cout << "cumAlive: " << cumAlive_u << endl;
                        //cout << "cumAlive_exp: " << cumAlive_exp_u << endl;


          }

        } // end of each diam class


        //            // debug stuff on vHa
        //            cout << regId << "|" << ft << "|cumTp_temp|";
        //            for (uint u=0; u<dClasses.size(); u++){
        //                cout << cumTp_temp.at(u)<<"|";
        //            }
        //            cout << endl;
        //            cout << regId << "|" << ft << "|cumTp_exp|";
        //            for (uint u=0; u<dClasses.size(); u++){
        //                cout << cumTp_exp_temp.at(u)<<"|";
        //            }
        //            cout << endl;
        //            cout << regId << "|" << ft << "|vHa_temp|";
        //            for (uint u=0; u<dClasses.size(); u++){
        //                cout << vHa_temp.at(u)<<"|";
        //            }
        //            cout << endl;
        //            cout << regId << "|" << ft << "|vHa_exp|";
        //            for (uint u=0; u<dClasses.size(); u++){
        //                cout << vHa_exp_temp.at(u)<<"|";
        //            }
        //            cout << endl;


      } // end of each ft
    } // end of each region
    MD->setErrorLevel(MSG_ERROR);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.25.3.5 void computeInventary ( )

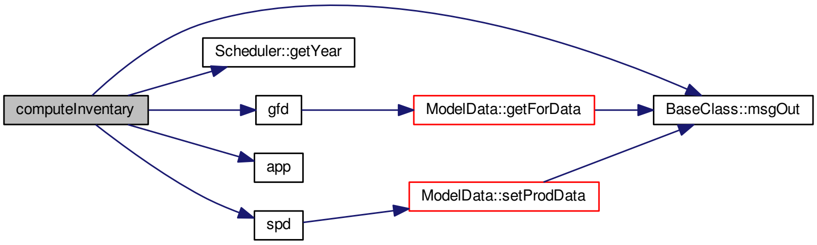
in=f(vol_t-1)

Definition at line 670 of file ModelCore.cpp.

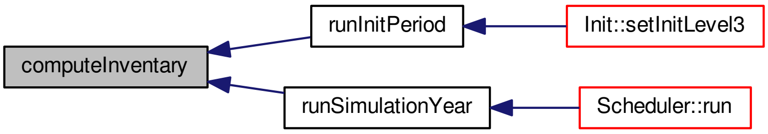
Referenced by runInitPeriod(), and runSimulationYear().

```
00670                           {
00671   msgOut(MSG_INFO, "Starting computing inventary available for this year..");
00672   int thisYear = MTHREAD->SCD->getYear();
00673 
00674   // In(i,p_pr,t)   =   sum((u,lambda,essence),prov(u,essence,lambda,p_pr)*V(u,i,lambda,essence,t-1));
00675   for(uint i=0;i<regIds2.size();i++){
00676     int r2 = regIds2[i];
00677     for(uint pp=0;pp<priProducts.size();pp++){
00678       double in = 0;
00679       for(uint ft=0;ft<fTypes.size();ft++){
00680         for(uint dc=0;dc<dClasses.size();dc++){
00681           double vol = dc?gfd("vol",r2,fTypes[ft],dClasses[dc],thisYear-1):0.;
00682           in += app(priProducts[pp],fTypes[ft],dClasses[dc])*vol;
00683         }
00684       }
00685       spd(in,"in",r2,priProducts[pp],thisYear,true);
00686 
00687     }
00688   } // end of for each region
00689 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



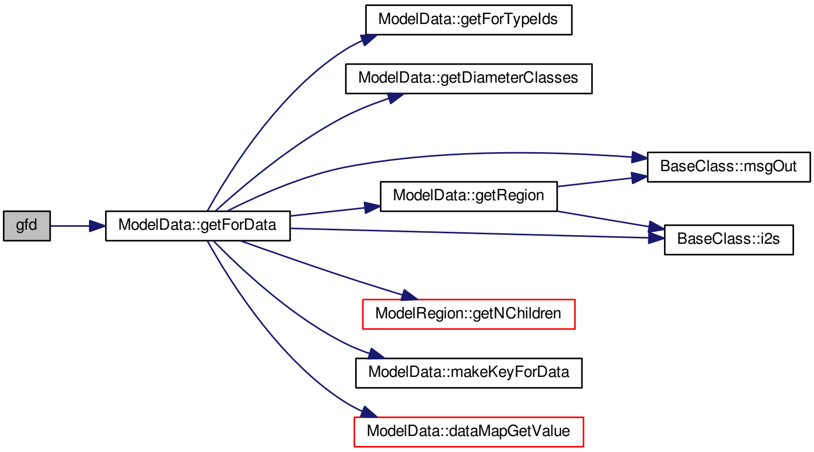
4.25.3.6 double gfd (const string & *type_h,* const int & *regId_h,* const string & *forType_h,* const string & *freeDim_h,* const int & *year* = DATA_NOW) const [inline], [private]

Definition at line 67 of file ModelCore.h.

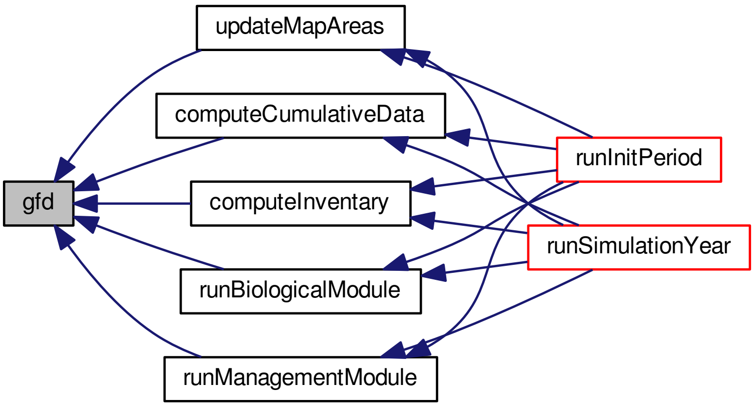
Referenced by computeCumulativeData(), computeInventary(), runBiologicalModule(), runManagementModule(), and updateMapAreas().

```
00067 {return MTHREAD->MD->getForData(type_h, regId_h, forType_h, freeDim_h, year);};
```

Here is the call graph for this function:


gfd
ModelData::getForData
ModelData::getForTypeIds
ModelData::getDiameterClasses
BaseClass::msgOut
ModelData::getRegion
BaseClass::i2s
ModelRegion::getNChildren
ModelData::makeKeyForData
ModelData::dataMapGetValue

Here is the caller graph for this function:



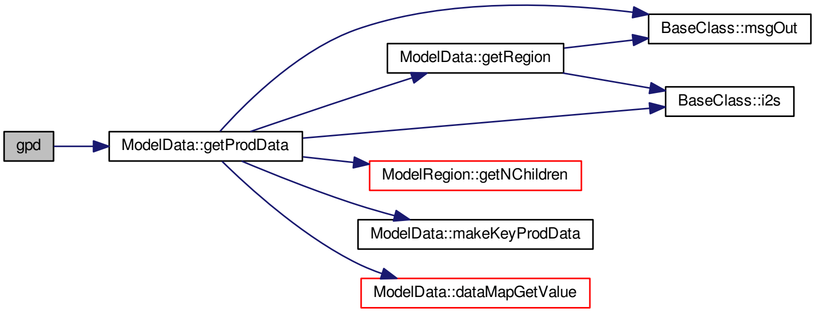
4.25.3.7 double gpd (const string & *type_h*, const int & *regId_h*, const string & *prodId_h*, const int & *year* = DATA_NOW, const string & *freeDim_h* = "") const `[inline]`, `[private]`

Definition at line 66 of file ModelCore.h.

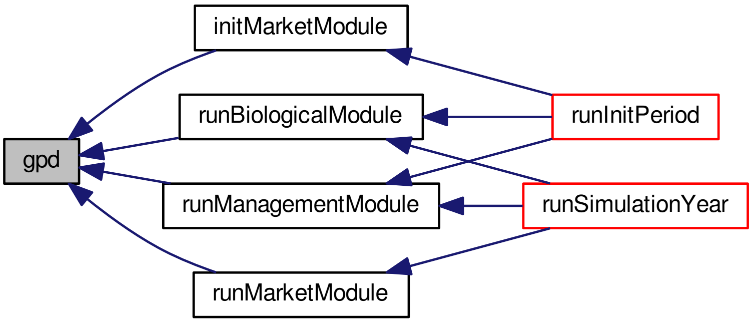
Referenced by initMarketModule(), runBiologicalModule(), runManagementModule(), and runMarketModule().

```
00066 {return MTHREAD->MD->getProdData(type_h, regId_h, prodId_h, year, freeDim_h);};
```

Here is the call graph for this function:


gpd
ModelData::getProdData
BaseClass::msgOut
ModelData::getRegion
BaseClass::i2s
ModelRegion::getNChildren
ModelData::makeKeyProdData
ModelData::dataMapGetValue

Here is the caller graph for this function:



4.25.3.8 void initMarketModule ( )

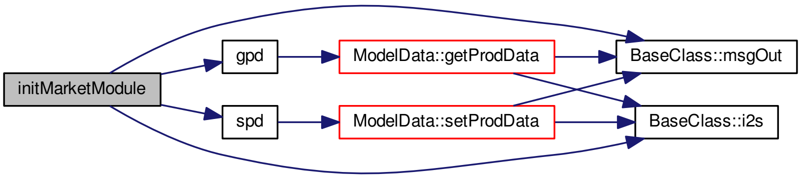
computes st and pw for second year and several needed-only-at-t0-vars for the market module

Definition at line 93 of file ModelCore.cpp.

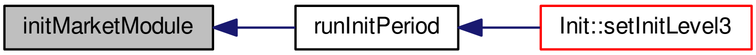
Referenced by runInitPeriod().

```
00093                              {
00094   msgOut(MSG_INFO, "Starting market module (init stage)..");
00095   for(uint i=0;i<regIds2.size();i++){
00096     int r2 = regIds2[i];
00097 
00098     //RPAR('pl',i,p_tr,t-1)    =  sum(p_pr, a(p_pr,p_tr)*RPAR('pl',i,p_pr,t-1))+m(i,p_tr);
00099     for(uint sp=0;sp<secProducts.size();sp++){
00100       double value = 0;
00101       for (uint pp=0;pp<priProducts.size();pp++){
00102         value += gpd("pl",r2,priProducts[pp],secondYear)*
00103              gpd("a",r2,priProducts[pp],secondYear,
      secProducts[sp]);
00104       }
00105       value += gpd("m",r2,secProducts[sp],secondYear);
00106       spd(value,"pl",r2,secProducts[sp],secondYear);
00107     }
00108     // RPAR('dl',i,p_pr,t-1)    =  sum(p_tr, a(p_pr,p_tr)*RPAR('sl',i,p_tr,t-1));
00109     for (uint pp=0;pp<priProducts.size();pp++){
00110       double value=0;
00111       for(uint sp=0;sp<secProducts.size();sp++){
00112         value += gpd("sl",r2,secProducts[sp],secondYear)*
00113              gpd("a",r2,priProducts[pp],secondYear,
      secProducts[sp]);
00114       }
00115       spd(value,"dl",r2,priProducts[pp],secondYear,true);
00116     }
00117     // RPAR('st',i,prd,t-1)     =  RPAR('sl',i,prd,t-1)+RPAR('sa',i,prd,t-1);
00118     // RPAR('dt',i,prd,t-1)     =  RPAR('dl',i,prd,t-1)+RPAR('da',i,prd,t-1);
00119     for (uint ap=0;ap<allProducts.size();ap++){
00120       double stvalue =   gpd("sl",r2,allProducts[ap],secondYear)
00121               + gpd("sa",r2,allProducts[ap],secondYear);
00122       double dtvalue =   gpd("dl",r2,allProducts[ap],secondYear)
00123               + gpd("da",r2,allProducts[ap],secondYear);
00124       spd(stvalue,"st",r2,allProducts[ap],secondYear,true);
00125       spd(dtvalue,"dt",r2,allProducts[ap],secondYear,true);
00126     }
00127 
00128     // q1(i,p_tr)  =
       1/(1+((RPAR('dl',i,p_tr,t-1)/RPAR('da',i,p_tr,t-1))**(1/psi(i,p_tr)))*(RPAR('pl',i,p_tr,t-1)/PT(p_tr,t-1)));
00129     // p1(i,p_tr)                  =  1-q1(i,p_tr);
00130     // RPAR('dc',i,p_tr,t-1)    =  (q1(i,p_tr)*RPAR('da',i,p_tr,t-1)**((psi(i,p_tr)-1)/psi(i,p_tr))+
       p1(i,p_tr)*RPAR('dl',i,p_tr,t-1)**((psi(i,p_tr)-1)/psi(i,p_tr)))**(psi(i,p_tr)/(psi(i,p_tr)-1));
00131     // RPAR('pc',i,p_tr,t-1)    =
(RPAR(’da’,i,p_tr,t-1)/RPAR(’dc’,i,p_tr,t-1))*PT(p_tr,t-1)+(RPAR(’dl’,i,p_tr,t-1)/RPAR(’dc’,i,p_tr,t-1))*RPAR(’pl’,i,p_
    // RPAR(’pc’,i,p_pr,t-1)    = (RPAR(’sa’,i,p_pr,t-1)/RPAR(’sc’,i,p_pr,t-1))*PT(p_pr,t-1)+(RPAR(’sl’,i,p_pr,t-1)/RPAR(’sc’,i,p_pr,t-1))*RPAR(’pl’,i,p_
    // RPAR(’pw’,i,p_tr,t-1)    = (RPAR(’dl’,i,p_tr,t-1)*RPAR(’pl’,i,p_tr,t-1)+RPAR(’da’,i,p_tr,t-1)*PT(p_tr,t-1))/RPAR(’dt’,i,p_tr,t-1) ; //changed 201
    // K(i,p_tr,t-1)            =  k1(i,p_tr)*RPAR(’sl’,i,p_tr,t-1);
    for(uint sp=0;sp<secProducts.size();sp++){
      double psi = gpd("psi",r2,secProducts[sp],secondYear);
      double dl  = gpd("dl",r2,secProducts[sp],secondYear);
      double da  = gpd("da",r2,secProducts[sp],secondYear);
      double pl  = gpd("pl",r2,secProducts[sp],secondYear);
      double sl  = gpd("sl",r2,secProducts[sp],secondYear);
      double k1  = gpd("k1",r2,secProducts[sp],secondYear);
      double pWo = gpd("pl",WL2,secProducts[sp],secondYear); // World price (local price for region 99999)


      double q1  = 1/ ( 1+pow(dl/da,1/psi)*(pl/pWo) );
      double p1  = 1-q1;
      double dc  = pow(
               q1*pow(da,(psi-1)/psi) + p1*pow(dl,(psi-1)/psi)
              ,
               psi/(psi-1)
              );
      double pc = (da/dc)*pWo
              +(dl/dc)*pl;
      double pw = (dl*pl+da*pWo)/(dl+da);
      double k = k1*sl;

      spd(q1,"q1",r2,secProducts[sp],firstYear,true);
      spd(p1,"p1",r2,secProducts[sp],firstYear,true);
      spd(dc,"dc",r2,secProducts[sp],secondYear,true);
      spd(pc,"pc",r2,secProducts[sp],secondYear,true);
      spd(pw,"pw",r2,secProducts[sp],secondYear,true);
      spd(k,"k",r2,secProducts[sp],secondYear,true);
    }

    // t1(i,p_pr)               = 1/(1+((RPAR(’sl’,i,p_pr,t-1)/RPAR(’sa’,i,p_pr,t-1))**(1/eta(i,p_pr)))*(RPAR(’pl’,i,p_pr,t-1)/PT(p_pr,t-1)));
    // r1(i,p_pr)               =  1-t1(i,p_pr);
    // RPAR(’sc’,i,p_pr,t-1)    =  (t1(i,p_pr)*RPAR(’sa’,i,p_pr,t-1)**((eta(i,p_pr)-1)/eta(i,p_pr))+ r1(i,p_pr)*RPAR(’sl’,i,p_pr,t-1)**((eta(i,p_pr)-1)/eta(i,p_pr)))**(eta(i,p_pr)/(eta(i,p_pr)-1))
    // RPAR(’pc’,i,p_pr,t-1)    = (RPAR(’sa’,i,p_pr,t-1)/RPAR(’sc’,i,p_pr,t-1))*PT(p_pr,t-1)+(RPAR(’sl’,i,p_pr,t-1)/RPAR(’sc’,i,p_pr,t-1))*RPAR(’pl’,i,p_
    // RPAR(’pw’,i,p_pr,t-1)    = (RPAR(’sl’,i,p_pr,t-1)*RPAR(’pl’,i,p_pr,t-1)+RPAR(’sa’,i,p_pr,t-1)*PT(p_pr,t-1))/RPAR(’st’,i,p_pr,t-1) ; //changed 201
    for(uint pp=0;pp<priProducts.size();pp++){

      double sl  = gpd("sl",r2,priProducts[pp],secondYear);
      double sa  = gpd("sa",r2,priProducts[pp],secondYear);
      double eta = gpd("eta",r2,priProducts[pp],secondYear);
      double pl  = gpd("pl",r2,priProducts[pp],secondYear);
      double pWo = gpd("pl",WL2,priProducts[pp],secondYear); // World price (local price for region 99999)


      double t1 = 1/ ( 1+(pow(sl/sa,1/eta))*(pl/pWo) );
      double r1 = 1-t1;
      double sc = pow(
                t1*pow(sa,(eta-1)/eta) + r1*pow(sl,(eta-1)/eta)
             ,
                eta/(eta-1)
             );
      double pc = (sa/sc)*pWo+(sl/sc)*pl;
      double pw = (sl*pl+sa*pWo)/(sl+sa);

      spd(t1,"t1",r2,priProducts[pp],firstYear,true);
      spd(r1,"r1",r2,priProducts[pp],firstYear,true);
      spd(sc,"sc",r2,priProducts[pp],secondYear,true);
      spd(pc,"pc",r2,priProducts[pp],secondYear,true);
      spd(pw,"pw",r2,priProducts[pp],secondYear,true);
    }
  } // end of each region


  // initializing the exports to zero quantities
  for(uint r1=0;r1<l2r.size();r1++){
    for(uint r2=0;r2<l2r[r1].size();r2++){
      for(uint p=0;p<allProducts.size();p++){
       for(uint r2To=0;r2To<l2r[r1].size();r2To++){
          spd(0,"rt",l2r[r1][r2],allProducts[p],secondYear,true,i2s(l2r[r1][r2To]));  // regional trade, it was exp in gams
        }
      }
    }
  } // end of r1 region
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.25.3.9 void runBiologicalModule ( )

computes hV, hArea and new vol at end of year

Todo Harvest volumes from death trees

Definition at line 363 of file ModelCore.cpp.

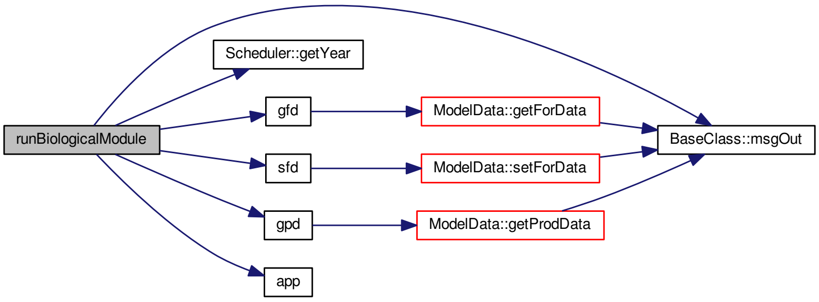
Referenced by runInitPeriod(), and runSimulationYear().

```
00363                                {
00364 
00365   msgOut(MSG_INFO, "Starting resource module..");
00366   hV_byPrd.clear();
00367   int thisYear = MTHREAD->SCD->getYear();
00368   int previousYear = thisYear-1;
00369 
00370   for(uint i=0;i<regIds2.size();i++){
00371 
00372     int r2 = regIds2[i];
00373     int regId = r2;
00374     // Post optimisation biological mobule..
00375     vector < vector < vector <double> > > hV_byPrd_regional;
00376     for(uint j=0;j<fTypes.size();j++){
00377       string ft = fTypes[j];
00378       vector < vector <double> > hV_byPrd_ft;
00379 
00380       // calculating the regeneration..
00381       // if we are in a year where the time of passage has not yet been reached
00382       // for the specific i,e,l then we use the exogenous Vregen, otherwise we
00383       // calculate it
00384       //if ( not scen("fxVreg") ,
00385       //  loop( (i,essence,lambda),
00386       //    if( ord(t)>=(tp_u1(i,essence,lambda)+2),
00387       //
     Vregen(i,lambda,essence,t)=regArea(i,essence,lambda,t-tp_u1(i,essence,lambda))*volHa_u1(i,essence,lambda)/1000000    ;
00388       //    );
00389       //  );
```

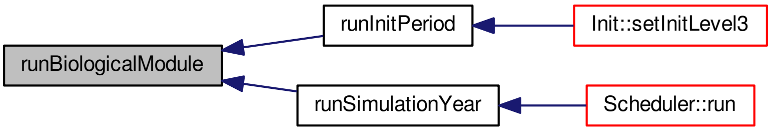
```
     //);
           int tp_u0 = gfd("tp",regId,ft,dClasses[0]); // time of passage to reach the first
diameter class // bug 20140318, added ceil
     if(regType != "fixed" && (thisYear-secondYear) >= tp_u0 ) { // T.O.D.O to be checked
-> 20121109 OK
       double pastRegArea = gfd("regArea",regId,ft,"",thisYear-tp_u0);
       double vHa = gfd("vHa",regId,ft,dClasses[1]);
       //cout << "vHa - entryVolHa: " << vHa << " / " << entryVolHa << endl;
       double vReg = pastRegArea*vHa/1000000; // T.O.D.O: check the 1000000. -> Should be ok, as area in
ha vol in Mm^3
       sfd(vReg,"vReg",regId,ft,"");
     }

     for (uint u=0; u<dClasses.size(); u++){
       string dc = dClasses[u];
       double hr = 0;
       double pastYearVol = u?gfd("vol",regId,ft,dc,previousYear):0.;
       double hV_mort = 0.; /// \todo Harvest volumes from death trees
       vector <double> hV_byPrd_dc;

       // harvesting rate & volumes...
       //hr(u,i,essence,lambda,t) =    sum(p_pr,
prov(u,essence,lambda,p_pr)*RPAR('st',i,p_pr,t)/In(i,p_pr,t));
       //hV(u,i,essence,lambda,t) = hr(u,i,essence,lambda,t) * V(u,i,lambda,essence,t-1);
       //hV_byPrd(u,i,essence,lambda,p_pr,t) =
prov(u,essence,lambda,p_pr)*(RPAR('st',i,p_pr,t)/In(i,p_pr,t))*V(u,i,lambda,essence,t-1);
               //double debug =0;
       for(uint pp=0;pp<priProducts.size();pp++){
         double st = gpd("st",regId,priProducts[pp]);
         double in = gpd("in",regId,priProducts[pp]);
         double hr_pr = u?app(priProducts[pp],ft,dc)*st/ in:0;
         double hV_byPrd_dc_prd = hr_pr*pastYearVol;
         hr += hr_pr;
         hV_byPrd_dc.push_back( hV_byPrd_dc_prd);
                   //debug += hV_byPrd_dc_prd;
       }
       double hV = hr*pastYearVol;
               //double debug2 = debug-hV;

               // test passed 20131203
               //if(debug2  < -0.000000000001 || debug2  > 0.000000000001){
               //    cout << "Problems!" << endl;
               //}

       // post harvesting remained volumes computation..
       // loop(u$(ord(u)=1),
       //  first diameter class, no harvesting and fixed regenaration..
       //  V(u,i,lambda,essence,t)=(1-1/(tp(u,i,lambda,essence))-mort(u,i,lambda,essence)
)*V(u,i,lambda,essence,t-1)
       //                         +Vregen(i,lambda,essence,t);
       // );
       // loop(u$(ord(u)>1),
       //  generic case..
       //  V(u,i,lambda,essence,t)=((1-1/(tp(u,i,lambda,essence))
       //                         -mort(u,i,lambda,essence) -
hr(u,i,essence,lambda,t))*V(u,i,lambda,essence,t-1)
       //
+(1/(tp(u-1,i,lambda,essence)))*beta(u,i,lambda,essence)*V(u-1,i,lambda,essence,t-1));
       double vol;
       double newMortVol;    // new mortality volumes
       double tp          = gfd("tp",regId,ft,dc);
       double mort        = u?gfd("mortCoef",regId,ft,dc):0.;
       double vReg        = gfd("vReg",regId,ft,""); // Taking it from the memory database as we could
be in a fixed vReg scenario and not having calculated it from above!
       double beta        = u?gfd("betaCoef",regId,ft,dc):0.;
       //double hv2fa       = gfd("hv2fa",regId,ft,dc);
       double vHa         = gfd("vHa",regId,ft,dc);
       double finalHarvestFlag = gfd("finalHarvestFlag",regId,ft,dc);

       if(u==0){
         vol = 0.;
       } else if(u==1){
         vol = (1-1/tp-mort)*pastYearVol+vReg;
         newMortVol = mort*pastYearVol;
         double debug = vol;
       } else {
         double inc = (u==dClasses.size()-1)?0:1./tp; // we exclude the possibility for trees in
the last diameter class to move to an upper class
         double tp_1 = gfd("tp",regId,ft,dClasses[u-1]);
         double pastYearVol_1 = gfd("vol",regId,ft,dClasses[u-1],previousYear);
         vol = (1-inc-mort-hr)*pastYearVol+(1/tp_1)*beta*pastYearVol_1;
         newMortVol = mort*pastYearVol;
         double debug = vol;
       }
       double freeArea_byU = u?finalHarvestFlag*1000000*hV/vHa:0; // volumes are in Mm^3, area in ha, vHa
in m^3/ha
       //double debug = hv2fa*hr*pastYearVol*100;
```

```
    //cout << "regId|ft|dc| debug | freeArea: " << r2 << "|"<<ft<<"|"<<dc<<"| "<< debug << " | " << freeArea_byU << endl;

    sfd(hr,"hr",regId,ft,dc,DATA_NOW,true);
    sfd(hV,"hV",regId,ft,dc,DATA_NOW,true);
    sfd(vol,"vol",regId,ft,dc,DATA_NOW,true); // allowCreate needed for u==0
    sfd(newMortVol,"mortV",regId,ft,dc,DATA_NOW,true);

    sfd(freeArea_byU,"harvestedArea",regId,ft,dc,DATA_NOW, true);
    hV_byPrd_ft.push_back(hV_byPrd_dc);
   } // end foreach diameter classes
   hV_byPrd_regional.push_back(hV_byPrd_ft);
  } // end of each forest type
  hV_byPrd.push_back(hV_byPrd_regional);
 } // end of for each region

}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.25.3.10 void runInitPeriod ( )

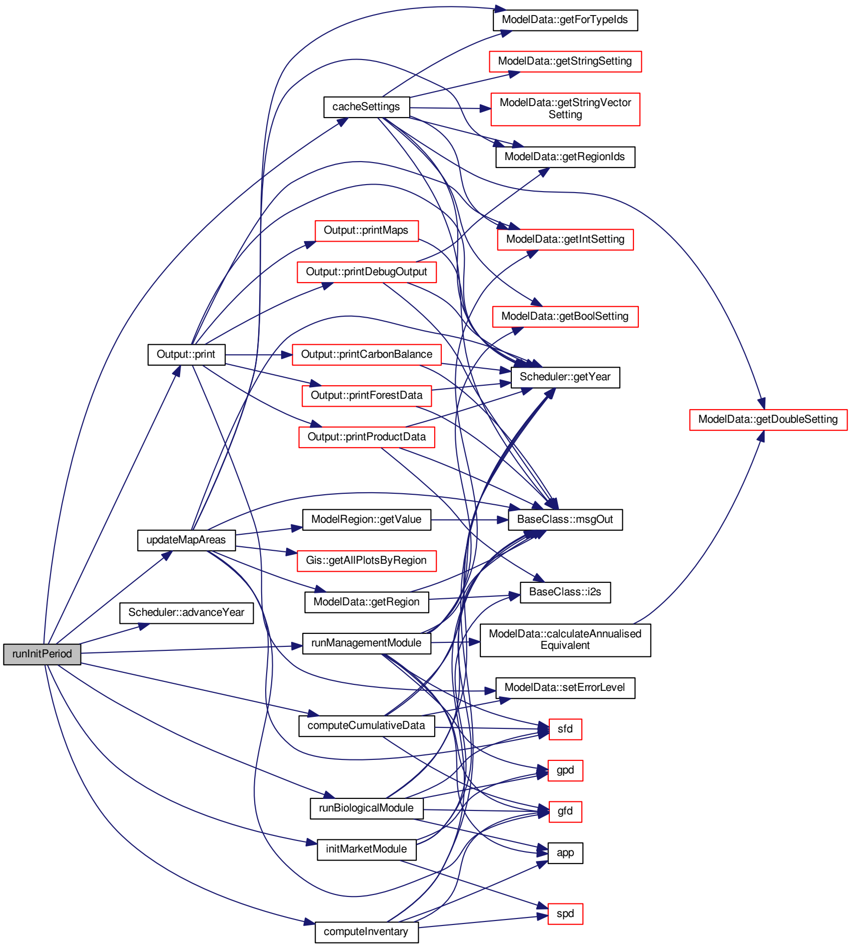
IMPORTANT NOTE: Volumes in Mm^3, Areas in the model in Ha (10000 m^2), in the layers in m^2 Some importan notes: V (volumes) -> at the end of the year In (inventary) -> at the beginning of the year Volumes are in Mm^3, Areas in the model in Ha (10000 m^2), in the layers in m^2

Definition at line 50 of file ModelCore.cpp.

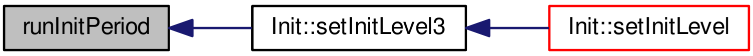
Referenced by Init::setInitLevel3().

```
                         {
  /**
  Some importan notes:
  V (volumes) -> at the end of the year
  In (inventary) -> at the beginning of the year
  Volumes are in Mm^3, Areas in the model in Ha (10000 m^2), in the layers in m^2
  */
  cacheSettings();              // cashe things like first year, second year, dClasses...
  initMarketModule();           // inside it uses first year, second year
  MTHREAD->DO->print();
  MTHREAD->SCD->advanceYear(); // 2005->2006
  computeInventary();           // in=f(vol_t-1)
  computeCumulativeData();      // compute cumTp_exp, vHa_exp, vHa
  runBiologicalModule();        //
  runManagementModule();
  updateMapAreas();             // update the forArea_{ft} layer on each pixel as old
 value-hArea+regArea
  MTHREAD->DO->print();
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.25.3.11 void runManagementModule ( )

computes regArea and expectedReturns

see ::calculateAnnualisedEquivalent

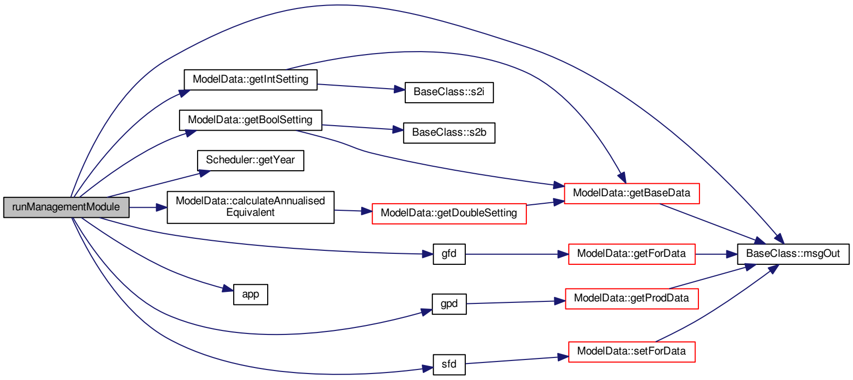
Definition at line 484 of file ModelCore.cpp.

Referenced by runInitPeriod(), and runSimulationYear().

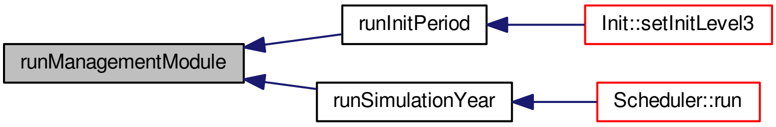
```
00484                              {
00485
00486  msgOut(MSG_INFO, "Starting management module..");
00487    //int thisYear = MTHREAD->SCD->getYear();
00488    //int previousYear = thisYear-1;
00489    MTHREAD->DO->expReturnsDebug.clear();
00490    int outputLevel = MTHREAD->MD->getIntSetting("outputLevel");
00491    bool weightedAverageExpectedReturns = MTHREAD->MD->getBoolSetting("
      weightedAverageExpectedReturns");
00492
00493    //vector <vector < vector <vector <vector <double> > > > > expReturnsDebug; ///< l2_region, for type,
       d.c., pr prod, variable name
00494    //cout << "year/dc/pp/eai/cumTp/vHa/pw" << endl;
00495
00496    int thisYear = MTHREAD->SCD->getYear();
00497
00498  for(uint i=0;i<regIds2.size();i++){
00499        vector < vector <vector <vector <double> > > >  expReturnsDebug_region;
00500
00501    int r2 = regIds2[i];
00502    int regId = r2;
00503    vector <double> cachedExpectedReturnsByFt;
00504
00505    // PART 1: COMPUTING THE EXPECTED RETURNS FOR EACH FOREST TYPE
00506
00507    for(uint j=0;j<fTypes.size();j++){
00508      string ft = fTypes[j];
00509            vector <vector <vector <double> > >  expReturnsDebug_ft;
00510      // Post optimisation management mobule..
00511
00512      //if(regType != "fixed" && regType != "fromHrLevel"){
00513      // 20120910, Antonello: changed.. calculating the expected returns also for fixed and fromHrLevel
       regeneration (then not used but gives indication)
00514      // calculating the expected returns..
00515      //  loop ( (u,i,essence,lambda,p_pr),
00516      //    if (sum(u2, hV(u2,i,essence,lambda,t))= 0,
00517      //      expRetPondCoef(u,i,essence,lambda,p_pr) = 0;
00518      //    else
00519      //      expRetPondCoef(u,i,essence,lambda,p_pr) = hV_byPrd(u,i,essence,lambda,p_pr,t)/ sum(u2,
       hV(u2,i,essence,lambda,t));
00520      //    );
00521      //  );
00522      //  expReturns(i,essence,lambda) = sum( (u,p_pr),
00523      //            RPAR("pl",i,p_pr,t)*hv2fa(i,essence,lambda,u)*(1/df_byFT(u,i,lambda,essence))*
       // df_byFT(u,i,lambda,essence)
00524      //            expRetPondCoef(u,i,essence,lambda,p_pr)
00525      //            );
00526      double hV_byFT = 0.; // gfd("hV",regId,ft,DIAM_PROD); // it must include only final harvested
       products in order to act as weightering agent
00527      double expReturns = 0;
00528
00529
```

```
      for (uint u=0; u<dClasses.size(); u++){
        string dc = dClasses[u];
        double finalHarvestFlag = gfd("finalHarvestFlag",regId,ft,dc);
                double hV = gfd("hV",regId,ft,dc);
                hV_byFT += finalHarvestFlag * hV;
      }

            if(hV_byFT==0. || !weightedAverageExpectedReturns){ // nothing has been harvested in this pixel for this specific forest type. Let's calculate the combination product/diameter class with the highest expected return
        for (uint u=0; u<dClasses.size(); u++){
                    vector <vector <double> >  expReturnsDebug_dc;
          string dc = dClasses[u];
          double vHa              = gfd("vHa_exp",regId,ft,dc);
          double finalHarvestFlag = gfd("finalHarvestFlag",regId,ft,dc);
          double cumTp_u          = gfd("cumTp_exp",regId,ft,dc);
          for (uint pp=0;pp<priProducts.size();pp++){
                        vector <double>  expReturnsDebug_pp;
            double pw    = gpd("pw",regId,priProducts[pp]);
            double raw_amount = finalHarvestFlag*pw*vHa*app(priProducts[pp],ft,dc); // B.U.G. 20121126, it was missing app(pp,ft,dc) !!
            double anualised_amount =  MD->calculateAnnualisedEquivalent(raw_amount,cumTp_u);
                        if (anualised_amount>expReturns) {
                            expReturns=anualised_amount;
                           // if (ft == "con_highF" && regId == 11041){
                           //     cout << thisYear << "/" << dc << "/" << priProducts[pp] << "/" << anualised_amount << "/" << cumTp_u << "/" << vHa << "/" << pw << endl;
                           // }
                        }
                        if(outputLevel >= OUTVL_ALL){
                          expReturnsDebug_pp.push_back(0.0);
                          expReturnsDebug_pp.push_back(hV_byFT);
                          expReturnsDebug_pp.push_back(finalHarvestFlag);
                          expReturnsDebug_pp.push_back(0.0);
                          expReturnsDebug_pp.push_back(pw);
                          expReturnsDebug_pp.push_back(cumTp_u);
                          expReturnsDebug_pp.push_back(vHa);
                          expReturnsDebug_pp.push_back(anualised_amount);
                          expReturnsDebug_pp.push_back(0);
                        }
                        expReturnsDebug_dc.push_back(expReturnsDebug_pp);
                    } // end each pp
                    expReturnsDebug_ft.push_back(expReturnsDebug_dc);
                } // end dc
      } else {
        for (uint u=0; u<dClasses.size(); u++){
                    vector <vector <double> >  expReturnsDebug_dc;
          string dc  = dClasses[u];
          double vHa = gfd("vHa_exp",regId,ft,dc);
          double finalHarvestFlag = gfd("finalHarvestFlag",regId,ft,dc);
          double cumTp_u = gfd("cumTp_exp",regId,ft,dc);

          for (uint pp=0;pp<priProducts.size();pp++){
                        vector <double>  expReturnsDebug_pp;
            double pw    = gpd("pw",regId,priProducts[pp]);
            double pl    = gpd("pl",regId,priProducts[pp]);
            double pwor    = gpd("pl",99999,priProducts[pp]);

            double hVol_byUPp = hV_byPrd.at(i).at(j).at(u).at(pp); // harvested volumes for this product, diameter class on this region and forest type

            //double raw_amount_old = pw*hv2fa*    hVol_byUPp/hV_byFT; // old and wrong. it was in €/m^4
                        double raw_amount = finalHarvestFlag*pw*vHa* hVol_byUPp/hV_byFT; // now in €/ha if there is also thining volumes in hV_byFT, I underestimate expected returns !! TO.DO change it !! DONE, DONE...
            /**
            see @ModelData::calculateAnnualisedEquivalent
            */
            double anualised_amount =  MD->calculateAnnualisedEquivalent(raw_amount,cumTp_u); //comTp is on diamClasses, u here is on pDiamClasses
            //cout << "reg|ft|dc|prd|raw amount|ann.amount|tp|hV|hVTot|pw|pl|pW|vHa|fHFlag;";
            //cout << regId <<";"<< ft <<";"<< dc <<";" << priProducts[pp] <<";" << raw_amount <<";"<< anualised_amount<<";";
            //cout << cumTp_u <<";"<< hVol_byUPp << ";" << hV_byFT << ";" << pw << ";" << pl << ";" << pwor << ";" << vHa << ";" << finalHarvestFlag << endl;
            expReturns += anualised_amount;

                        if(outputLevel >= OUTVL_ALL){
                          expReturnsDebug_pp.push_back(hVol_byUPp);
                          expReturnsDebug_pp.push_back(hV_byFT);
                          expReturnsDebug_pp.push_back(finalHarvestFlag);
                          expReturnsDebug_pp.push_back(finalHarvestFlag*hVol_byUPp/hV_byFT);
                          expReturnsDebug_pp.push_back(pw);
                          expReturnsDebug_pp.push_back(cumTp_u);
                          expReturnsDebug_pp.push_back(vHa);
                          expReturnsDebug_pp.push_back(MD->
calculateAnnualisedEquivalent(finalHarvestFlag*pw*vHa,cumTp_u));
                          expReturnsDebug_pp.push_back(1);
                        }
                        expReturnsDebug_dc.push_back(expReturnsDebug_pp);
                    } // end for each priProducts

                    expReturnsDebug_ft.push_back(expReturnsDebug_dc);
          //expReturnsPondCoef.push_back(expReturnsPondCoef_byPrd);
                } // end for each dc
            } // ending "it has been harvested" condition
      double debug = expReturns;
      sfd(expReturns,"expReturns",regId, ft,"",DATA_NOW,true);
      cachedExpectedReturnsByFt.push_back(expReturns);
            expReturnsDebug_region.push_back(expReturnsDebug_ft);
        } // end foreach forest
        MTHREAD->DO->expReturnsDebug.push_back(expReturnsDebug_region);


    // PART 2: ALLOCATING THE HARVESTED AREA TO REGENERATION AREA BASED ON EXPECTED RETURNS

    // calculating freeArea at the end of the year and choosing the new regeneration area..
    //freeArea(i,essence,lambda) = sum(u, hv2fa(i,essence,lambda,u)*hr(u,i,essence,lambda,t)*V(u,i,lambda,essence,t-1)*100);
    //if(scen("endVreg") ,
    //  regArea(i,essence,lambda,t) = freeArea(i,essence, lambda);   // here we could introduce in/out area from other land usages
    //else
    //  loop (i,
    //    loop( (essence,lambda),
    //      if ( expReturns(i,essence,lambda) = smax( (essence2,lambda2),expReturns(i,essence2,lambda2) ),
    //        regArea (i,essence,lambda,t) =  sum( (essence2, lambda2), freeArea(i,essence2, lambda2) ) * mr;
    //      );
    //    );
    //    regArea(i,essence,lambda,t) = freeArea(i,essence, lambda)*(1-mr);   // here we could introduce in/out area from other land usages
    //  );
    double totalHarvestedArea = gfd("harvestedArea",regId,FT_ALL,DIAM_ALL);

    double maxExpReturns = *( max_element( cachedExpectedReturnsByFt.begin(), cachedExpectedReturnsByFt.end() ) );
    bool foundMaxExpReturns = false;
    for(uint j=0;j<fTypes.size();j++){
      string ft = fTypes[j];
      double harvestedAreaForThisFT = gfd("harvestedArea",regId,ft,DIAM_ALL);
      if(regType == "fixed" || regType == "fromHrLevel"){
        // regeneration area is the harvested area..
        double harvestedArea = harvestedAreaForThisFT;
        sfd(harvestedArea,"regArea",regId,ft,"",DATA_NOW,true);
      } else {
        // regeneration area is a mix between harvested area and the harvested area of te most profitting forest type..
        double regArea = 0;
        if (!foundMaxExpReturns && cachedExpectedReturnsByFt[j] == maxExpReturns){
                    // I use the foundMaxExpReturns for the unlikely event that two forest types have the same expected return to avoid overcounting of the area
          regArea += totalHarvestedArea*mr;
          foundMaxExpReturns = true;
        }
                double freq = rescaleFrequencies ? gfd("freq_norm",regId,ft,""):gfd("freq",regId,ft,""); // "probability of presence" for unmanaged forest, added 20140318
                regArea +=  harvestedAreaForThisFT*(1-mr)*freq;
        sfd(regArea,"regArea",regId,ft,"",DATA_NOW,true);
      }
    }// end of foreach forest type
    double totalRegArea = gfd("regArea",regId,FT_ALL,DIAM_ALL);
    } // end of each r2
    //vector <vector < vector <vector <vector <double> > > > > expReturnsDebug = MTHREAD->DO->expReturnsDebug;
    //cout << "bla" << endl;

}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.25.3.12 void runMarketModule ( )

computes st (supply total) and pw (weighted price). Optimisation inside.

Definition at line 211 of file ModelCore.cpp.

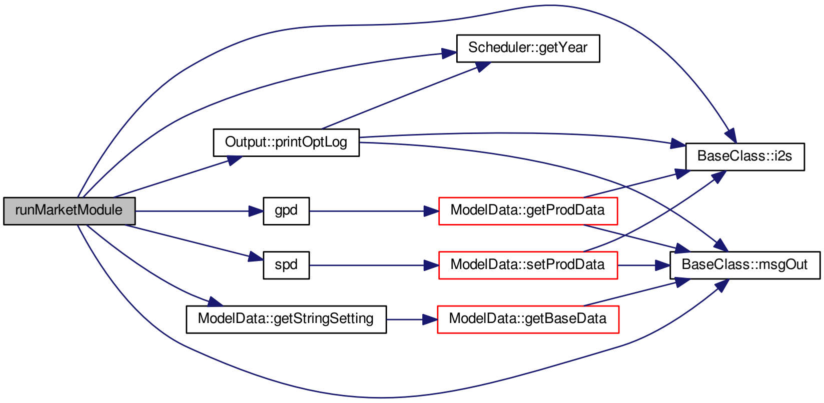
Referenced by runSimulationYear().

```
00211                           {
00212 
00213   // *** PRE-OPTIMISATION YEARLY OPERATIONS..
00214 
00215   // Updating variables
00216   // notes:
00217   // - dispo_sup: not actually entering anywhere, forgiving it for now..
00218   // - dc0: not needed, it is just the first year demand composite
00219   int thisYear = MTHREAD->SCD->getYear();
00220   int previousYear = thisYear-1;
00221 
00222   for(uint i=0;i<regIds2.size();i++){
00223     int r2 = regIds2[i];
00224 
00225     // K(i,p_tr,t)  =  (1+g1(i,p_tr))*K(i,p_tr,t-1);
00226     // AA(i,p_tr)    =
       (sigma(p_tr)/(sigma(p_tr)+1))*RPAR(’pc’,i,p_tr,t-1)*(RPAR(’dc’,i,p_tr,t-1)**(-1/sigma(p_tr)));
00227     // GG(i,p_tr)    =    RPAR(’dc’,i,p_tr,t-1)*((RPAR(’pc’,i,p_tr,t-1))**(-sigma(p_tr))); //alpha
00228     for(uint sp=0;sp<secProducts.size();sp++){
00229       double g1    = gpd("g1",r2,secProducts[sp],previousYear);
00230       double sigma = gpd("sigma",r2,secProducts[sp]);
00231       double pc_1  = gpd("pc",r2,secProducts[sp],previousYear);
00232       double dc_1  = gpd("dc",r2,secProducts[sp],previousYear);
00233       double k_1   = gpd("k",r2,secProducts[sp],previousYear);
```

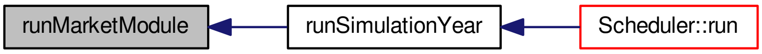
```

      double k  = (1+g1)*k_1;
      double aa = (sigma/(sigma+1))*pc_1*pow(dc_1,-1/sigma);
      double gg = dc_1*pow(pc_1,-sigma); //alpha

      spd(k, "k" ,r2,secProducts[sp]);
      spd(aa,"aa",r2,secProducts[sp],DATA_NOW,true);
      spd(gg,"gg",r2,secProducts[sp],DATA_NOW,true);
    }

    // BB(i,p_pr)   = (sigma(p_pr)/(sigma(p_pr)+1))*RPAR('pc',i,p_pr,t-1)*(RPAR('sc',i,p_pr,t-1)**(-1/sigma(p_pr)))*(In(i,p_pr,t-1)/In(i,p_p
    // FF(i,p_pr)   = RPAR('sc',i,p_pr,t-1)*((RPAR('pc',i,p_pr,t-1))**(-sigma(p_pr)))*(In(i,p_pr,t)/In(i,p_pr,t-1))**(gamma(p_pr)); //chi
    for(uint pp=0;pp<priProducts.size();pp++){
      double gamma = gpd("gamma",r2,priProducts[pp]);
      double sigma = gpd("sigma",r2,priProducts[pp]);
      double pc_1  = gpd("pc",r2,priProducts[pp],previousYear);
      double sc_1  = gpd("sc",r2,priProducts[pp],previousYear);
      double in    = gpd("in",r2,priProducts[pp]);
      double in_1  = gpd("in",r2,priProducts[pp],previousYear);

      double bb = (sigma/(sigma+1))*pc_1*pow(sc_1,-1/sigma)*pow(in_1/in,gamma/sigma);
      double ff = sc_1*pow(pc_1,-sigma)*pow(in/in_1,gamma); //chi

      spd(bb,"bb",r2,priProducts[pp],DATA_NOW,true);
      spd(ff,"ff",r2,priProducts[pp],DATA_NOW,true);
    }

  } // end for each region in level 2 (and updating variables)

  // *** OPTIMISATION....

  // Create an instance of the IpoptApplication
  //Opt *OPTa = new Opt(MTHREAD);
  //SmartPtr<TNLP> OPTa = new Opt(MTHREAD);
  //int initialOptYear = MTHREAD->MD->getIntSetting("initialOptYear");
  SmartPtr<IpoptApplication> application = new IpoptApplication();
  //if(thisYear == initialOptYear){
    //application = new IpoptApplication();
  //} else {
  //  application->Options()->SetStringValue("warm_start_init_point", "yes");
  //}
  string linearSolver = MTHREAD->MD->getStringSetting("linearSolver");
  application->Options()->SetStringValue("linear_solver", linearSolver); // default in ipopt is ma27
  //application->Options()->SetStringValue("hessian_approximation", "limited-memory"); // quasi-newton approximation of the hessian
  //application->Options()->SetIntegerValue("mumps_mem_percent", 100);
  application->Options()->SetNumericValue("obj_scaling_factor", -1); // maximisation
  application->Options()->SetNumericValue("max_cpu_time", 1800); // max 1/2 hour to find the optimus for one single year
  application->Options()->SetStringValue("check_derivatives_for_naninf", "yes"); // detect error but may crash the application.. TO.DO catch this error!
  //application->Options()->SetStringValue("nlp_scaling_method", "equilibration-based"); // much worster
  // Initialize the IpoptApplication and process the options
  ApplicationReturnStatus status;
  status = application->Initialize();
  if (status != Solve_Succeeded) {
    printf("\n\n*** Error during initialization!\n");
    msgOut(MSG_INFO,"Error during initialization! Do you have the solver compiled for the specified linear solver?");
    return;
  }


  msgOut(MSG_INFO,"Running optimisation problem for this year (it may take a few minutes for large models)..");
  status = application->OptimizeTNLP(MTHREAD->OPT);

  // *** POST OPTIMISATION....

  // post-equilibrium variables->parameters assignments..
  // RPAR(type,i,prd,t)   =  RVAR.l(type,i,prd);
  // EX(i,j,prd,t)        =  EXP.l(i,j,prd);
  // ObjT(t)              =  Obj.l ;
  // ==> in Opt::finalize_solution()

  // Retrieve some statistics about the solve
  if (status == Solve_Succeeded) {
    Index iter_count = application->Statistics()->IterationCount();
    Number final_obj = application->Statistics()->FinalObjective();
    printf("\n*** The problem solved in %d iterations!\n", iter_count);
    printf("\n*** The final value of the objective function is %e.\n", final_obj);
    msgOut(MSG_INFO, "The problem solved successfully in "+i2s(iter_count)+" iterations.");
    int icount = iter_count;
    double obj = final_obj;
```

```
00313     MTHREAD->DO->printOptLog(true, icount, obj);
00314   } else {
00315     //Number final_obj = application->Statistics()->FinalObjective();
00316     cout << "***ERROR: MODEL DIDN'T SOLVE FOR THIS YEAR" << endl;
00317         msgOut(MSG_CRITICAL_ERROR, "Model DIDN'T SOLVE for this year");
00318     // IMPORTANT! Don't place the next two lines above the msgOut() function or it will crash in windows if the user press the stop button
00319         //Index iter_count = application->Statistics()->IterationCount(); // sys error if model didn't solve
00320         //Number final_obj = application->Statistics()->FinalObjective();
00321         int icount = 0;
00322         double obj = 0;
00323     MTHREAD->DO->printOptLog(false, icount, obj);
00324   }
00325 
00326   for(uint r2= 0; r2<regIds2.size();r2++){  // you can use r2<=regIds2.size() to try an out-of range memory error that is not detected other than by valgrind (with a message "Invalid read of size 4 in ModelCore::runSimulationYear() in src/ModelCore.cpp:351")
00327     int regId = regIds2[r2];
00328 
00329     //  // total supply and total demand..
00330     //  RPAR('st',i,prd,t)   =  RPAR('sl',i,prd,t)+RPAR('sa',i,prd,t);
00331     //  RPAR('dt',i,prd,t)   =  RPAR('dl',i,prd,t)+RPAR('da',i,prd,t);
00332     //  // weighted prices.. //changed 20120419
00333     //  RPAR('pw',i,p_tr,t)   =  (RPAR('dl',i,p_tr,t)*RPAR('pl',i,p_tr,t)+RPAR('da',i,p_tr,t)*PT(p_tr,t))/RPAR('dt',i,p_tr,t) ; //changed 20120419
00334     //  RPAR('pw',i,p_pr,t)   =  (RPAR('sl',i,p_pr,t)*RPAR('pl',i,p_pr,t)+RPAR('sa',i,p_pr,t)*PT(p_pr,t))/RPAR('st',i,p_pr,t) ; //changed 20120419
00335     for (uint p=0;p<allProducts.size();p++){
00336       double st = gpd("sl",regId,allProducts[p])+gpd("sa",regId, allProducts[p]);
00337       double dt = gpd("dl",regId,allProducts[p])+gpd("da",regId, allProducts[p]);
00338       spd(st,"st",regId,allProducts[p]);
00339       spd(dt,"dt",regId,allProducts[p]);
00340     }
00341     for (uint p=0;p<secProducts.size();p++){
00342       double dl = gpd("dl",regId,secProducts[p]);
00343       double pl = gpd("pl",regId,secProducts[p]);
00344       double da = gpd("da",regId,secProducts[p]); // bug corrected 20120913
00345       double pworld = gpd("pl", WL2,secProducts[p]);
00346       double dt = gpd("dt",regId,secProducts[p]);
00347       double pw = (dl*pl+da*pworld)/dt;
00348       spd(pw,"pw",regId,secProducts[p]);
00349     }
00350     for (uint p=0;p<priProducts.size();p++){
00351       double sl = gpd("sl",regId,priProducts[p]);
00352       double pl = gpd("pl",regId,priProducts[p]);
00353       double sa = gpd("sa",regId,priProducts[p]);  // bug corrected 20120913
00354       double pworld = gpd("pl", WL2,priProducts[p]);
00355       double st = gpd("st",regId,priProducts[p]);
00356       double pw = (sl*pl+sa*pworld)/st;
00357       spd(pw,"pw",regId,priProducts[p]);
00358     }
00359   } // end of foreach region
00360 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



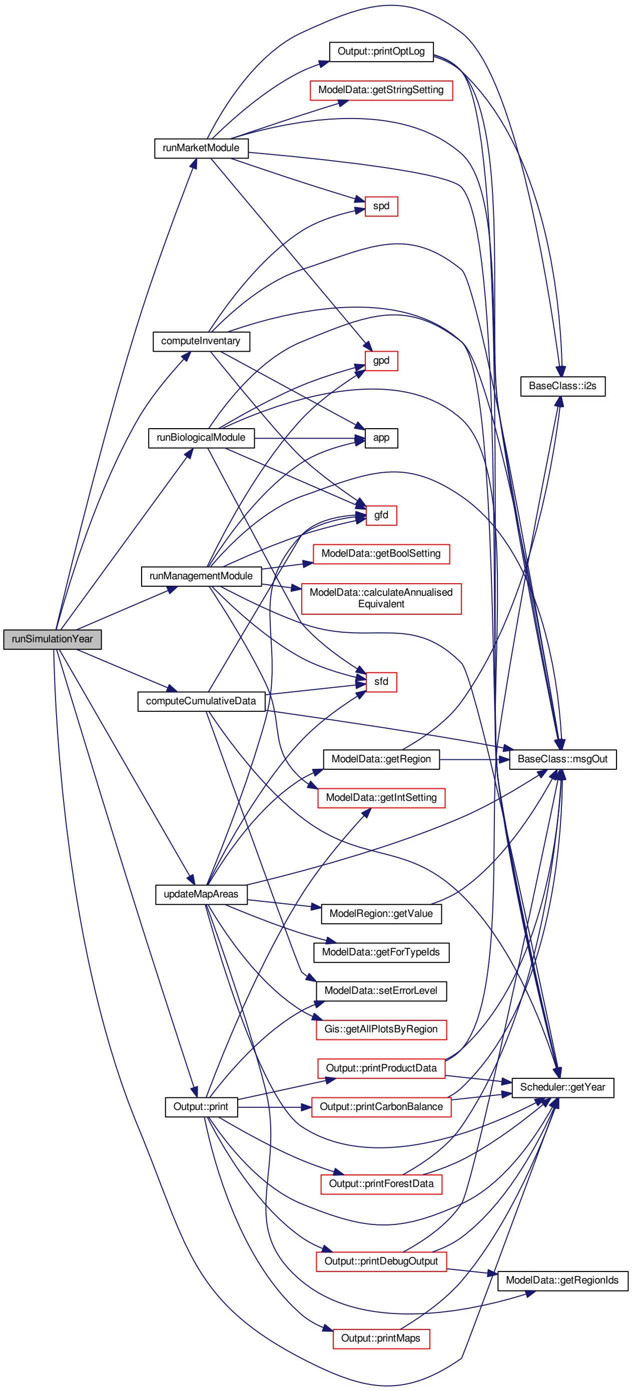
4.25.3.13 void runSimulationYear ( )

Definition at line 70 of file ModelCore.cpp.

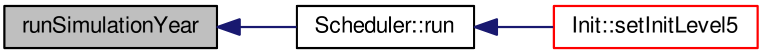
Referenced by Scheduler::run().

```
00070                            {
00071   int thisYear = MTHREAD->SCD->getYear();
00072   computeInventary();          // in=f(vol_t-1)
00073   runMarketModule();
00074   computeCumulativeData();     // compute cumTp_exp, vHa_exp
00075   runBiologicalModule();
00076
00077     /*double sl = gpd("sl",11041,'softWRoundW');
00078     double pl = gpd("pl",11041,'softWRoundW');
00079     double sa = gpd("sa",11041,'softWRoundW');
00080     double pworld = gpd("pl", WL2,'softWRoundW');
00081     double st = gpd("st",11041,'softWRoundW');
00082     double pw = (sl*pl+sa*pworld)/st;
00083     cout << thisYear <<
00084     */
00085
00086   runManagementModule();
00087   updateMapAreas();
00088   MTHREAD->DO->print();
00089 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



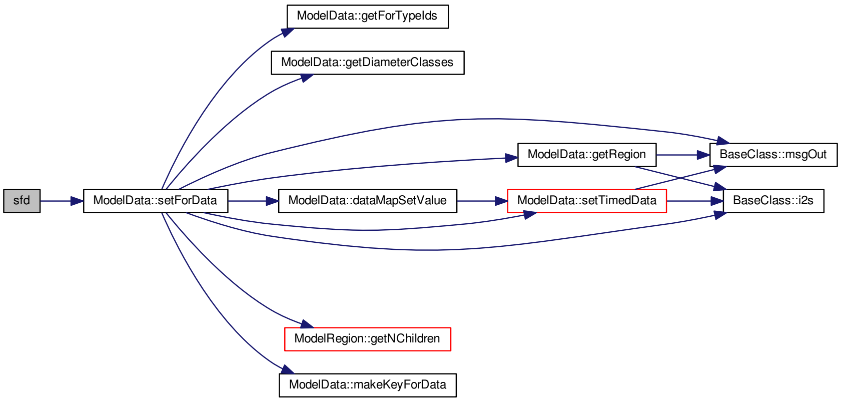
4.25.3.14 void sfd (const double & *value_h*, const string & *type_h*, const int & *regId_h*, const string & *forType_h*, const string & *freeDim_h*, const int & *year* = DATA_NOW, const bool & *allowCreate* = `false`) const `[inline]`, `[private]`

Definition at line 69 of file ModelCore.h.

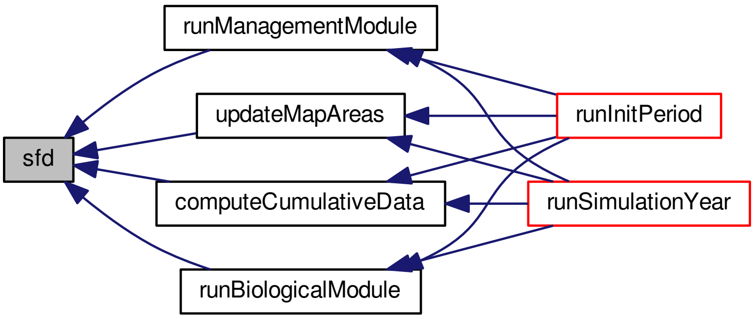
Referenced by computeCumulativeData(), runBiologicalModule(), runManagementModule(), and updateMapAreas().

```
00069 {MTHREAD->MD->setForData(value_h, type_h, regId_h, forType_h, freeDim_h, year,
      allowCreate);};
```

Here is the call graph for this function:



Here is the caller graph for this function:



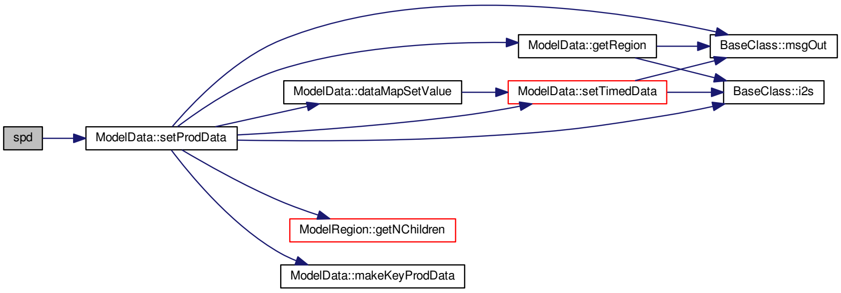
4.25.3.15 void spd (const double & *value_h*, const string & *type_h*, const int & *regId_h*, const string & *prodId_h*, const int & *year* = DATA_NOW, const bool & *allowCreate* = `false`, const string & *freeDim_h* = "") const `[inline]`, `[private]`

Definition at line 68 of file ModelCore.h.

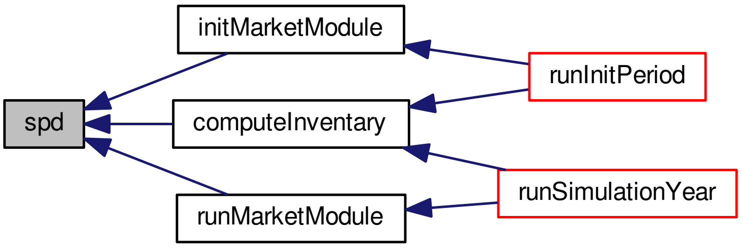
Referenced by computeInventary(), initMarketModule(), and runMarketModule().

```
00068 {MTHREAD->MD->setProdData(value_h, type_h, regId_h, prodId_h, year, allowCreate,
      freeDim_h);};
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.25.3.16 void updateMapAreas ( )

computes forArea_{ft}

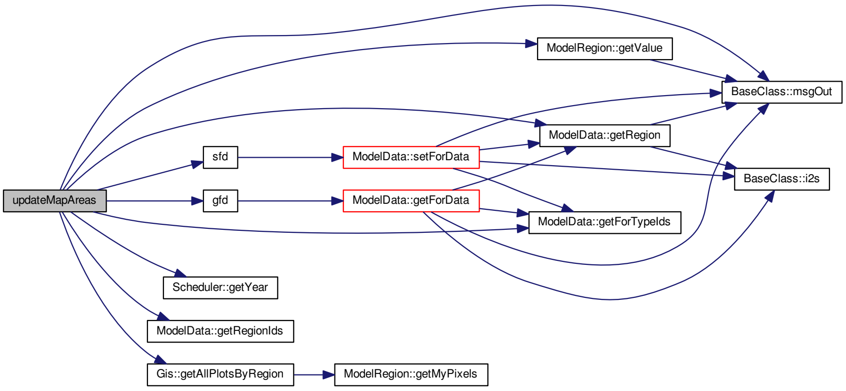
This function take for each region the difference for each forest type between the harvested area and the new regeneration one and apply such delta to each pixel of the region proportionally to the area that it already hosts.

Definition at line 895 of file ModelCore.cpp.

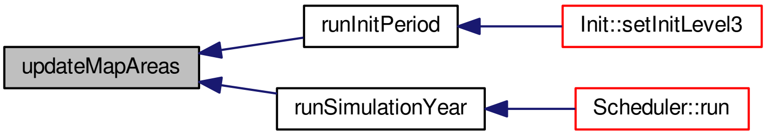
Referenced by runInitPeriod(), and runSimulationYear().

```
00895                          {
00896
00897   msgOut(MSG_INFO, "Updating map areas..");
00898   map<int,double> forestArea; // foresta area by each region
00899   pair<int,double > forestAreaPair;
00900     int thisYear = MTHREAD->SCD->getYear();
00901   vector<int> l2Regions =  MTHREAD->MD->getRegionIds(2, true);
00902   vector <string> fTypes =  MTHREAD->MD->getForTypeIds();
00903   int nFTypes = fTypes.size();
00904   int nL2Regions = l2Regions.size();
00905   for(uint i=0;i<nL2Regions;i++){
00906     int regId = l2Regions[i];
00907     vector<Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(regId);
00908     ModelRegion* reg = MTHREAD->MD->getRegion(regId);
00909     for(uint j=0;j<nFTypes;j++){
00910       string ft = fTypes[j];
00911             double oldRegioForArea;
00912             if(thisYear <= firstYear+1) {
00913                 oldRegioForArea = reg->getValue("forArea_"+ft)/10000;
00914             } else {
00915                 oldRegioForArea = gfd("forArea",regId,ft,DIAM_ALL,thisYear-1);
00916             }
00917             //oldRegioForArea = reg->getValue("forArea_"+ft)/10000;
00918             //double debug = gfd("forArea",regId,ft,DIAM_ALL,thisYear-1);
00919             //double debug_diff = oldRegioForArea - debug;
00920             //cout << thisYear << ";" << regId << ";" << ft << ";" << oldRegioForArea << ";" << debug <<
       ";" << debug_diff << endl;
00921             double harvestedArea = gfd("harvestedArea",regId,ft,DIAM_ALL); //20140206
00922             double regArea = gfd("regArea",regId,ft,DIAM_ALL); //20140206
00923       double newRegioForArea = oldRegioForArea + regArea - harvestedArea;
00924       sfd(newRegioForArea,"forArea",regId,ft,"",DATA_NOW, true);
00925       for(uint z=0;z<rpx.size();z++){
00926         double oldValue = rpx[z]->getDoubleValue("forArea_"+ft,true);
00927                 double ratio = newRegioForArea/oldRegioForArea;
00928                 double newValue = oldValue*ratio;
00929         rpx[z]->changeValue("forArea_"+ft, newValue);
00930       }
00931
00932     }
00933   }
00934 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.25.4 Member Data Documentation

4.25.4.1 vector<string> allProducts `[private]`

Definition at line 83 of file ModelCore.h.

Referenced by cacheSettings(), initMarketModule(), and runMarketModule().

4.25.4.2 vector<string> dClasses `[private]`

Definition at line 84 of file ModelCore.h.

Referenced by cacheSettings(), computeCumulativeData(), computeInventary(), runBiologicalModule(), and runManagementModule().

4.25.4.3 double expType `[private]`

Definition at line 89 of file ModelCore.h.

Referenced by cacheSettings(), and computeCumulativeData().

4.25.4.4 int firstYear [private]

Definition at line 76 of file ModelCore.h.

Referenced by cacheSettings(), computeCumulativeData(), initMarketModule(), and updateMapAreas().

4.25.4.5 vector<string> fTypes [private]

Definition at line 86 of file ModelCore.h.

Referenced by cacheSettings(), computeCumulativeData(), computeInventary(), runBiologicalModule(), runManagementModule(), and updateMapAreas().

4.25.4.6 vector< vector < vector < vector <double> > > > hV_byPrd [private]

Definition at line 91 of file ModelCore.h.

Referenced by runBiologicalModule(), and runManagementModule().

4.25.4.7 vector<vector <int> > l2r [private]

Definition at line 87 of file ModelCore.h.

Referenced by cacheSettings(), and initMarketModule().

4.25.4.8 ModelData* MD [private]

Definition at line 70 of file ModelCore.h.

Referenced by cacheSettings(), computeCumulativeData(), and runManagementModule().

4.25.4.9 double mr [private]

Definition at line 90 of file ModelCore.h.

Referenced by cacheSettings(), and runManagementModule().

4.25.4.10 vector<string> pDClasses [private]

Definition at line 85 of file ModelCore.h.

Referenced by cacheSettings().

4.25.4.11 vector<string> priProducts [private]

Definition at line 81 of file ModelCore.h.

Referenced by cacheSettings(), computeInventary(), initMarketModule(), runBiologicalModule(), runManagementModule(), and runMarketModule().

4.25.4.12 vector< int > regIds2 `[private]`

Definition at line 80 of file ModelCore.h.

Referenced by cacheSettings(), computeCumulativeData(), computeInventary(), initMarketModule(), runBiologicalModule(), runManagementModule(), and runMarketModule().

4.25.4.13 string regType `[private]`

Definition at line 88 of file ModelCore.h.

Referenced by cacheSettings(), runBiologicalModule(), and runManagementModule().

4.25.4.14 bool rescaleFrequencies `[private]`

Definition at line 93 of file ModelCore.h.

Referenced by cacheSettings(), and runManagementModule().

4.25.4.15 int secondYear `[private]`

Definition at line 77 of file ModelCore.h.

Referenced by cacheSettings(), initMarketModule(), and runBiologicalModule().

4.25.4.16 vector< string > secProducts `[private]`

Definition at line 82 of file ModelCore.h.

Referenced by cacheSettings(), initMarketModule(), and runMarketModule().

4.25.4.17 int thirdYear `[private]`

Definition at line 78 of file ModelCore.h.

Referenced by cacheSettings().

4.25.4.18 int WL2 `[private]`

Definition at line 79 of file ModelCore.h.

Referenced by cacheSettings(), initMarketModule(), and runMarketModule().

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

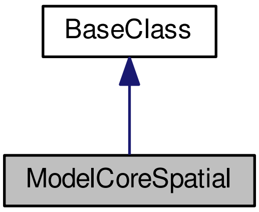
- /home/lobianco/git/ffsm_pp/src/ModelCore.h
- /home/lobianco/git/ffsm_pp/src/ModelCore.cpp

4.26 ModelCoreSpatial Class Reference

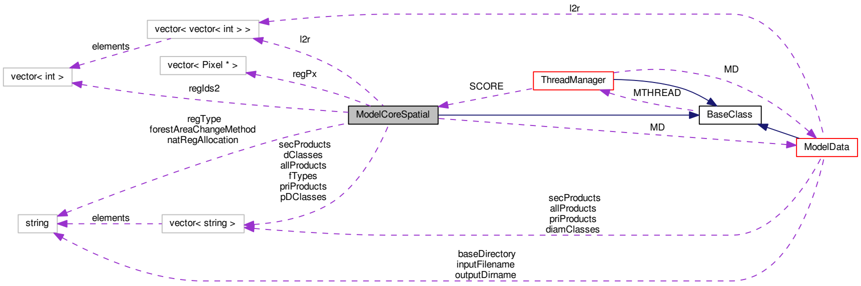
The core of the model (spatial version).

`#include <ModelCoreSpatial.h>`

Inheritance diagram for ModelCoreSpatial:



Collaboration diagram for ModelCoreSpatial:



Public Member Functions

- ModelCoreSpatial (ThreadManager ∗MTHREAD_h)
- ∼ModelCoreSpatial ()
- void runInitPeriod ()
- void runSimulationYear ()
- void initMarketModule ()

    computes st and pw for second year and several needed-only-at-t0-vars for the market module
- void runMarketModule ()

    computes st (supply total) and pw (weighted price). Optimisation inside.
- void runBiologicalModule ()

    computes hV, hArea and new vol at end of year
- void runManagementModule ()

    computes regArea and expectedReturns
- void sumRegionalForData ()

    computes vol, hV, harvestedArea, regArea and expReturns at reg level from the pixel level

- void initialiseCarbonModule ()

  call initialiseDeathBiomassStocks(), initialiseProductsStocks() and initialiseEmissionCounters()
- void initialiseDeathTimber ()

  Set deathTimberInventory to zero for the previous years (under the hipotesis that we don't have advanced stock of death biomass usable as timber at the beginning of the simulation)
- void registerCarbonEvents ()

  call registerHarvesting(), registerDeathBiomass(), registerProducts() and registerTransports()
- void cacheSettings ()

  just cache exogenous settings from ModelData
- void initializePixelVolumes ()

  distribuite regional exogenous volumes to pixel volumes using corine land cover area as weight
- void assignSpMultiplierPropToVols ()

  ModelCoreSpatial::assignSpMultiplierPropToVols assigns the spatial multiplier (used in the time of return) based no more on a Normal distribution but on the volumes present in the pixel: more volume, more the pixel is fit for the ft.
- void initializePixelArea ()

  compute px->area for each ft and dc
- void resetPixelValues ()

  swap volumes->lagged_volumes and reset the other pixel vectors
- void cachePixelExogenousData ()

  computes pixel level tp, meta and mort
- void computeInventary ()

  in=f(vol_t-1)
- void computeCumulativeData ()

  computes cumTp_exp, vHa_exp, vHa
- void updateMapAreas ()

  computes forArea_{ft}
- void updateOtherMapData ()

  update (if the layer exists) other gis-based data, as volumes and expected returns, taking them from the data in the px object
- double computeExpectedPrice (const double &curLocPrice, const double &worldCurPrice, const double &worldFutPrice, const double &sl, const double &sa, const double &expCoef)

  Compute weighted expected price for a given product.
- void printDebugInitRegionalValues ()

  print initial inv, st, sl and sa in each region
- vector< double > allocateHarvesting (vector< double > total_st, const int &regId)

  Using the deathTimberInventory map, this function allocate the total st in st from death timber (that goes reduce the deathTimberInventory map) and stFromHarvesting that is what it remains after the allocation to death timber.
- void loadExogenousForestLayers (const string &what)

  Set pixel volumes (what="vol") OR areas (what="area") by specific forest types as defined in gis layers for volumes and proportionally to volumes for areas.
- double gpd (const string &type_h, const int &regId_h, const string &prodId_h, const int &year=DATA_NOW, const string &freeDim_h="") const
- double gfd (const string &type_h, const int &regId_h, const string &forType_h, const string &freeDim_h, const int &year=DATA_NOW) const
- void spd (const double &value_h, const string &type_h, const int &regId_h, const string &prodId_h, const int &year=DATA_NOW, const bool &allowCreate=false, const string &freeDim_h="") const
- void sfd (const double &value_h, const string &type_h, const int &regId_h, const string &forType_h, const string &freeDim_h, const int &year=DATA_NOW, const bool &allowCreate=false) const
- bool app (const string &prod_h, const string &forType_h, const string &dClass_h) const

Private Attributes

- ModelData ∗ MD
- int firstYear
- int secondYear
- int thirdYear
- int WL2
- vector< int > regIds2
- vector< string > priProducts
- vector< string > secProducts
- vector< string > allProducts
- vector< string > dClasses
- vector< string > pDClasses
- vector< string > fTypes
- vector< vector< int > > l2r
- string regType
- string natRegAllocation
- vector< Pixel ∗ > regPx
- bool rescaleFrequencies
- bool oldVol2AreaMethod
- string forestAreaChangeMethod
- double ir

Additional Inherited Members

4.26.1 Detailed Description

The core of the model (spatial version).

Once the environment is initialised (mainly data load, space created), the model is run through the two functions runInitPeriod() and runSimulationYear().

Some importan notes: V (volumes) -> at the end of the year In (inventary) -> at the beginning of the year Area -> at the end of the year Harvesting -> at the beginning of the year Volumes are in Mm^3, Areas in the model in Ha (10000 m^2), in the layers in m^2, vHa in m^3/ha. Prices are in €/m^3.

BALANCE: PROD_forLocal (sl) + PROD_forExp (sa) + IMP (da) + sum_reg(reg_trade_in) = CONS_fromLocal (dl) + CONS_fromImp (da) + EXP (sa) + sum_reg(reg_trade_out) note that this means that sl includes alread reg_trade_out, and dl includes already reg_trade_in

Where are volumes information ?

- ip px->vol - by px, ft and dc
- in forDataMap (through gft()) - by reg, ft and dc Where is area information ?
- in px->area - by px, ft and dc
- in forDataMap (through gft()) - by reg, ft and dc
- in px->values map (forArea_∗ layer, through px->getDoubleValue()) - by px and ft

Aggregation of the Expected returns

The problem is how to aggregate the expected returns, given at pixel anf ft level, first at the regional level, then at the ft group level (B/C) and total forest level and finally at national level from regional one.

A - From pixel to region

- weighted by total forest area in the pixel B1 - From ft to ft group
- in each pixel we take the highest expRet within the pixel and we weight by farea to get the regional value B2 - From ft group to forest
- actually, from ft to group: like b1, but we take the highest value in each px for any ft and we weight by forest area in the px to get the regional value C - From region to country
- we weight the individual ft, ft group and forest by the different regional total forest areas.∗

Definition at line 82 of file ModelCoreSpatial.h.

4.26.2 Constructor & Destructor Documentation

4.26.2.1 ModelCoreSpatial (ThreadManager ∗ *MTHREAD_h*)

Definition at line 37 of file ModelCoreSpatial.cpp.

```
00037                                                         {
00038   MTHREAD = MTHREAD_h;
00039 }
```

4.26.2.2 ∼ModelCoreSpatial ( )

Definition at line 41 of file ModelCoreSpatial.cpp.

```
00041                                 {
00042
00043 }
```

4.26.3 Member Function Documentation

4.26.3.1 vector< double > allocateHarvesting (vector< double > *total_st,* const int & *regId*)

Using the deathTimberInventory map, this function allocate the total st in st from death timber (that goes reduce the deathTimberInventory map) and stFromHarvesting that is what it remains after the allocation to death timber.

ModelCoreSpatial::allocateHarvesting.

Parameters

total_st	vector of total supply by primary products

Returns

a vector of the remaining supply that goes allocated to alive timber (that is, to harvesting)

The algorithm is such that is loops the deathTimberInventory map for each year (newer to older), dc (higher to smaller) and ft. It compute the primary products allocable from that combination and allocate the cell amount to decrease the total_st of that products in a proportional way to what still remain of the allocable products.

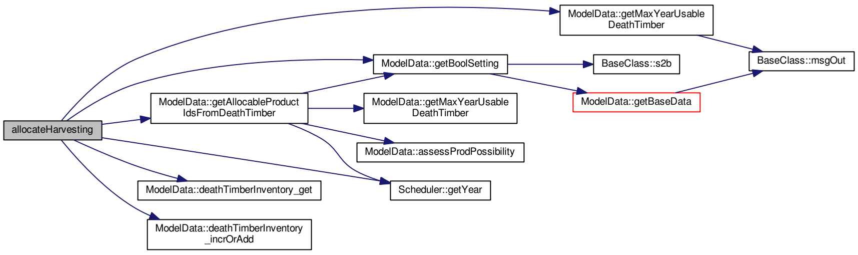
It is called in the runMarketModule() function.

Definition at line 2249 of file ModelCoreSpatial.cpp.

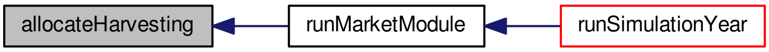
Referenced by runMarketModule().

```
02249                                                                                  {
02250    if(!MD->getBoolSetting("useDeathTimber")) return total_st;
02251    vector <double> stFromHarvesting(priProducts.size(),0.);
02252    //map<iisskey, double > *  deathTimberInventory= MD->getDeathTimberInventory();
02253    int maxYears = MD->getMaxYearUsableDeathTimber();
02254    int currentYear = MTHREAD->SCD->getYear();
02255    for(uint y = currentYear-1; y>currentYear-1-maxYears; y--){
02256      for (int u = dClasses.size()-1; u>=0; u--){  // I need to specify u as an integer !
02257        string dc = dClasses.at(u);
02258        for (uint f=0; f<fTypes.size(); f++){
02259          string ft = fTypes[f];
02260          vector<int>allocableProducts =  MD->
      getAllocableProductIdsFromDeathTimber(regId, ft, dc, y, currentYear-1)
      ;
02261          iisskey key(y,regId,ft,dc);
02262          double deathTimber = MD->deathTimberInventory_get(key);
02263          double sum_total_st_allocable = 0;
02264          // Computing shares/weights or remaining st to allcate
02265          for(uint ap=0;ap<allocableProducts.size();ap++){
02266            sum_total_st_allocable += total_st.at(allocableProducts[ap]);
02267          }
02268          for(uint ap=0;ap<allocableProducts.size();ap++){
02269            double allocableShare = sum_total_st_allocable?total_st.at(allocableProducts[ap])/
      sum_total_st_allocable:0.0;
02270            double allocated = min(total_st[allocableProducts[ap]],deathTimber*allocableShare);
02271            MD->deathTimberInventory_incrOrAdd(key,-allocated);
02272            total_st[allocableProducts[ap]] -= allocated;
02273          }
02274        }
02275      }
02276    }
02277    return total_st;
02278 }
```

Here is the call graph for this function:


ModelData::getMaxYearUsable
DeathTimber
ModelData::getBoolSetting
BaseClass::s2b
BaseClass::msgOut
ModelData::getAllocableProduct
IdsFromDeathTimber
ModelData::getMaxYearUsable
DeathTimber
ModelData::getBaseData
allocateHarvesting
ModelData::assessProdPossibility
ModelData::deathTimberInventory_get
Scheduler::getYear
ModelData::deathTimberInventory
_incrOrAdd

Here is the caller graph for this function:



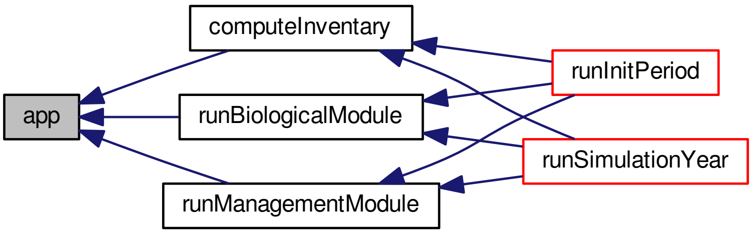
4.26.3.2 bool app (const string & *prod_h,* const string & *forType_h,* const string & *dClass_h*) const [inline]

Definition at line 120 of file ModelCoreSpatial.h.

Referenced by computeInventary(), runBiologicalModule(), and runManagementModule().

```
00120 {return MTHREAD->MD->assessProdPossibility(prod_h, forType_h, dClass_h);};
```

Here is the caller graph for this function:



4.26.3.3 void assignSpMultiplierPropToVols ( )

ModelCoreSpatial::assignSpMultiplierPropToVols assigns the spatial multiplier (used in the time of return) based no more on a Normal distribution but on the volumes present in the pixel: more volume, more the pixel is fit for the ft.

This function apply to the pixel a multiplier of time of passage that is inversely proportional to the volumes of that forest type present in the pixel. The idea is that in the spots where we observe more of a given forest type are probably the most suited ones to it.

The overall multipliers **of time of passage** (that is, the one returned by Pixel::getMultiplier("tp_multiplier")) will then be the product of this multiplier that account for spatial heterogeneity and of an eventual exogenous multiplier that accounts for different scenarios among the spatio-temporal dimensions.

Given that (forest type index omitted):

- V_p = volume of a given ft in each pixel (p)
- $\bar{g}$ and σ_g = regional average and standard deviation of the growth rate
- m_p = multiplier of time of passage

This multiplier is computed as:

- $v_p = max(V) - V_p$ A diff from the max volume is computed in each pixel
- $vr_p = v_p * \bar{g}/\bar{v}$ The volume diff is rescaled to match the regional growth rate
- $vrd_p = vr_p - \bar{vr}$ Deviation of the rescaled volumes are computed

- $vrdr_p = vrd_p * \sigma_g / \sigma_{vr}$ The deviations are then rescaled to match the standard deviations of the regional growth rate
- $m_p = (vrdr_p + \bar{vr})/\bar{g}$ The multiplier is computed from the ratio of the average rescaled volumes plus rescaled deviation over the average growth rate.

And it has the following properties:

- $\bar{m} = 1$
- $\sigma_m = cv_g$
- $m_p = V_p * \alpha + \beta$
- $m_{\bar{V}} = 1$

For spreadsheet "proof" see the file computation_of_growth_multipliers_from_know_avg_sd_and_proportional_↩ to_share_of_area_in_each_pixel.ods

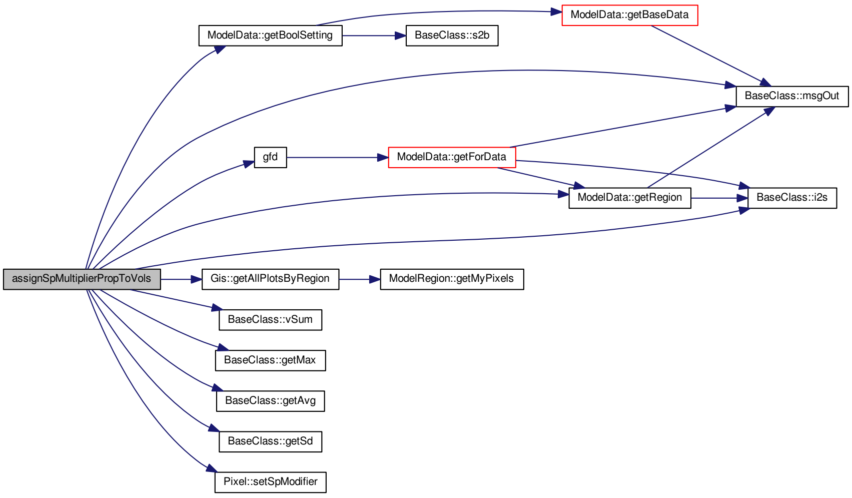
Definition at line 1114 of file ModelCoreSpatial.cpp.

Referenced by runInitPeriod().

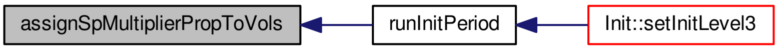
```
01114                                              {
01115
01116   if(!MTHREAD->MD->getBoolSetting("useSpatialVarPropToVol")){return;}
01117   for(uint r=0;r<regIds2.size();r++){
01118     int rId = regIds2[r];
01119     ModelRegion* reg = MD->getRegion(regIds2[r]);
01120     vector <Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(
      regIds2[r]);
01121     for(uint f=0;f<fTypes.size();f++){
01122       string ft = fTypes[f];
01123       double agr = gfd("agr",regIds2[r],ft,"");
01124       double sStDev = gfd("sStDev",regIds2[r],ft,"");
01125       vector<double> vols;
01126       vector<double> diffVols;
01127       vector<double> diffVols_rescaled;
01128       double diffVols_rescaled_deviation;
01129       double diffVols_rescaled_deviation_rescaled;
01130       double final_value;
01131       double multiplier;
01132       vector<double> multipliers; // for tests
01133
01134       double vol_max, rescale_ratio_avg, rescale_ratio_sd;
01135       double diffVols_avg, diffVols_rescaled_avg;
01136       double diffVols_rescaled_sd;
01137
01138       for (uint p=0;p<rpx.size();p++){
01139         Pixel* px = rpx[p];
01140         vols.push_back(vSum(px->vol[f]));
01141       } // end for each pixel
01142       vol_max=getMax(vols);
01143
01144       for(uint p=0;p<vols.size();p++){
01145         diffVols.push_back(vol_max-vols[p]);
01146       }
01147
01148       diffVols_avg = getAvg(diffVols);
01149       rescale_ratio_avg = (diffVols_avg != 0.0) ? agr/diffVols_avg : 1.0;
01150       for(uint p=0;p<diffVols.size();p++){
01151         diffVols_rescaled.push_back(diffVols[p]*rescale_ratio_avg);
01152       }
01153       diffVols_rescaled_avg = getAvg(diffVols_rescaled);
01154       diffVols_rescaled_sd  = getSd(diffVols_rescaled,false);
01155
01156       rescale_ratio_sd = (diffVols_rescaled_sd != 0.0) ? sStDev/diffVols_rescaled_sd : 1.0;
01157       for(uint p=0;p<diffVols_rescaled.size();p++){
01158         diffVols_rescaled_deviation          = diffVols_rescaled[p]     - diffVols_rescaled_avg;
01159         diffVols_rescaled_deviation_rescaled = diffVols_rescaled_deviation * rescale_ratio_sd;
01160         final_value                      = diffVols_rescaled_avg    + diffVols_rescaled_deviation_rescaled;
01161         multiplier = (agr != 0.0) ? min(1.6, max(0.4,final_value/agr)) : 1.0; //20151130: added bounds for
       extreme cases. Same bonds as in Gis::applySpatialStochasticValues()
01162          // multiplier = 1.0;
01163
```

```
        Pixel* px = rpx[p];
        px->setSpModifier(multiplier,f);
        multipliers.push_back(multiplier);
      }

      #ifdef QT_DEBUG
      // Check relaxed as we introduced bounds that may change slighly the avg and sd...
      double avgMultipliers = getAvg(multipliers);
      double sdMultipliers  = getSd(multipliers,false);
      if ( avgMultipliers < 0.9 || avgMultipliers > 1.1){
        msgOut(MSG_CRITICAL_ERROR, "The average of multipliers of ft "+ ft +" for
 the region " + i2s(rId) + " is not 1!");
      }
      if ( ( sdMultipliers -  (sStDev/agr) ) < -0.5 ||  ( sdMultipliers -  (sStDev/agr) ) > 0.5 ){
        double cv = sStDev/agr;
        msgOut(MSG_CRITICAL_ERROR, "The sd of multipliers of ft "+ ft +" for the
 region " + i2s(rId) + " is not equal to the spatial cv for the region!");
      }
      #endif
    } // end for each ft
  } // end for each region
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.26.3.4 void cachePixelExogenousData ( )

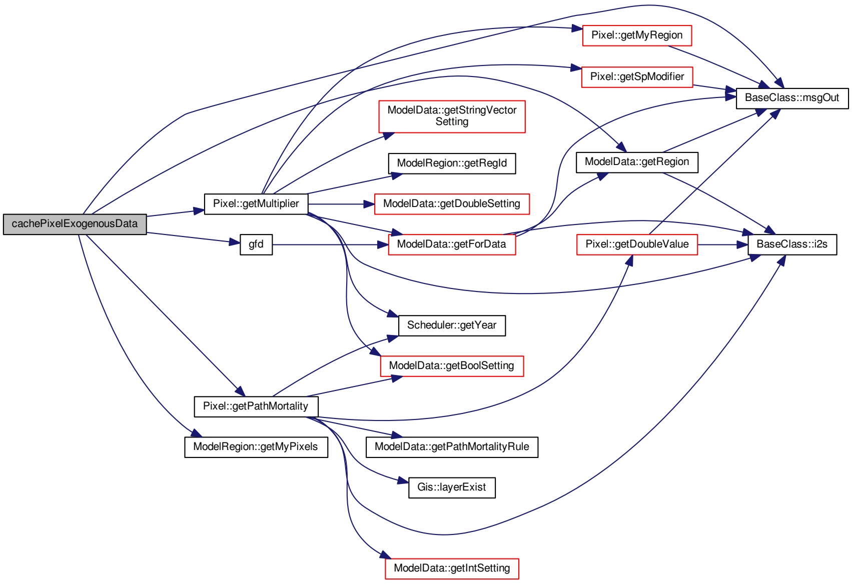
computes pixel level tp, meta and mort

Definition at line 1548 of file ModelCoreSpatial.cpp.

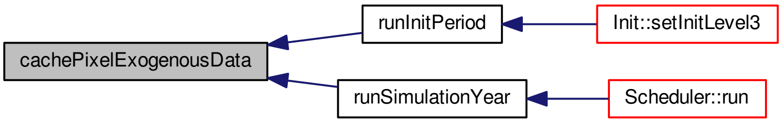
Referenced by runInitPeriod(), and runSimulationYear().

```
                                         {

  msgOut(MSG_INFO, "Starting cashing on pixel spatial-level exogenous data");
  for(uint r2= 0; r2<regIds2.size();r2++){
    int regId = regIds2[r2];
    regPx = MTHREAD->MD->getRegion(regId)->getMyPixels();
    for (uint p=0;p<regPx.size();p++){
      Pixel* px = regPx[p];
      px->tp.clear();
      px->beta.clear();
      px->mort.clear();

      for(uint j=0;j<fTypes.size();j++){
        string ft = fTypes[j];
        vector <double> tp_byu;
        vector <double> beta_byu;
        vector <double> mort_byu;

        double tp_multiplier_now       = px->getMultiplier("tp_multiplier",ft, DATA_NOW);
        double mortCoef_multiplier_now = px->getMultiplier("mortCoef_multiplier",ft, DATA_NOW);
        double betaCoef_multiplier_now = px->getMultiplier("betaCoef_multiplier",ft, DATA_NOW);


        for (uint u=0; u<dClasses.size(); u++){
          string dc = dClasses[u];
          double pathMortality           = px->getPathMortality(ft,dc, DATA_NOW);
          double tp, beta_real, mort_real;
          if (u==0){
            // tp of first diameter class not making it change across the time dimension, otherwise problems in getting the rigth past
            // regenerations. BUT good, px->tp.at(0) is used only to pick up the right regeneration, so the remaining of the model
            // uses the getMultiplier version and cumTp consider the dynamic effects also in the first dc.
            tp  = gfd("tp",regId,ft,dClasses[u],firstYear)*px->getMultiplier("tp_multiplier",ft,firstYear); // tp is defined also in the first diameter class, as it is the years to reach the NEXT diameter class
          } else {
            tp  = gfd("tp",regId,ft,dClasses[u],DATA_NOW)*tp_multiplier_now; // tp is defined also in the first diameter class, as it is the years to reach the NEXT diameter class
          }
          beta_real = u?gfd("betaCoef",regId,ft,dClasses[u],DATA_NOW)*betaCoef_multiplier_now:0;
          mort_real = min(u?gfd("mortCoef",regId,ft,dClasses[u], DATA_NOW)*mortCoef_multiplier_now+pathMortality :0,1.0);  //Antonello, bug fixed 20160203: In any case, natural plus pathogen mortality can not be larger than 1!
          tp_byu.push_back(tp);
          beta_byu.push_back(beta_real);
          mort_byu.push_back(mort_real);
        } // end of each tp
        px->tp.push_back(tp_byu);
        px->beta.push_back(beta_byu);
        px->mort.push_back(mort_byu);
      } // end of each ft
    } // end of each pixel
  } // end of each region
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.26.3.5 void cacheSettings ( )

just cache exogenous settings from ModelData

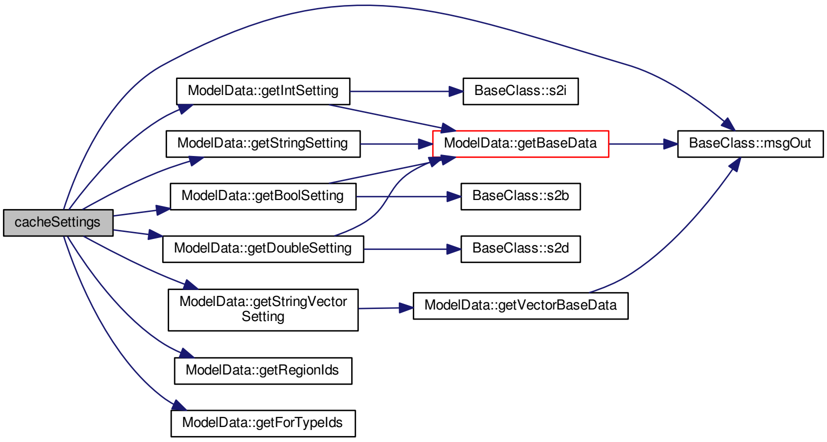
Definition at line 1021 of file ModelCoreSpatial.cpp.

Referenced by runInitPeriod().

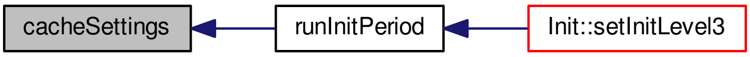
```
01021                                {
01022   msgOut(MSG_INFO, "Cashing initial model settings..");
01023   MD = MTHREAD->MD;
01024   firstYear = MD->getIntSetting("initialYear");
01025   secondYear = firstYear+1;
01026   thirdYear  = firstYear+2;
01027   WL2 = MD->getIntSetting("worldCodeLev2");
01028   regIds2 = MD->getRegionIds(2);
01029   priProducts = MD->getStringVectorSetting("priProducts");
01030   secProducts = MD->getStringVectorSetting("secProducts");
01031   allProducts = priProducts;
01032   allProducts.insert( allProducts.end(), secProducts.begin(),
      secProducts.end() );
01033   dClasses = MD->getStringVectorSetting("dClasses");
01034   pDClasses; // production diameter classes: exclude the fist diameter class below 15 cm
```

```
  pDClasses.insert(pDClasses.end(), dClasses.begin()+1, dClasses.end() );
  fTypes= MD->getForTypeIds();
  l2r = MD->getRegionIds();
  regType = MTHREAD->MD->getStringSetting("regType"); // how the regeneration should be computed (exogenous, from hr, from allocation choises)
  natRegAllocation = MTHREAD->MD->getStringSetting("natRegAllocation"); // how to allocate natural regeneration
  rescaleFrequencies = MD->getBoolSetting("rescaleFrequencies");
  oldVol2AreaMethod = MD->getBoolSetting("oldVol2AreaMethod");
  //mr = MD->getDoubleSetting("mr");
  forestAreaChangeMethod =  MTHREAD->MD->getStringSetting("forestAreaChangeMethod");
  ir = MD->getDoubleSetting("ir");


}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.26.3.6 void computeCumulativeData ( )

computes cumTp_exp, vHa_exp, vHa

Note on the effect of mortality modifiers on the entryVolHa. Unfortunatly for how it is defined the mortality multiplier (the ratio with the new mortality rate over the old one) we can't compute a entryVolHa based on it. It is NOT infact just like: vHa_adjusted = vHa_orig / mort_multiplier. The effect of mortality on the vHa of the first diameter class is unknow, and so we can't compute the effect of a relative increase.

param expType Specify how the forest owners (those that make the investments) behave will be the time of passage in the future in order to calculate the cumulative time of passage in turn used to discount future revenues. Will forest owners behave adaptively believing the time of passage between diameter classes will be like the observed one at time they make decision (0) or they will have full expectations believing forecasts (1) or something in the middle ? For compatibility with the GAMS code, a -1 value means using initial simulation tp values (fixed cumTp)."

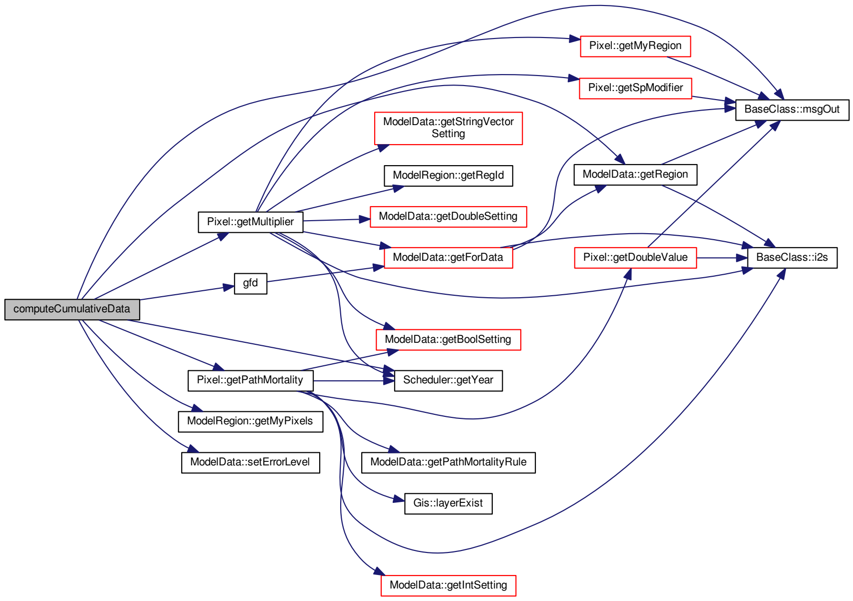
Definition at line 1312 of file ModelCoreSpatial.cpp.

Referenced by runInitPeriod(), and runSimulationYear().

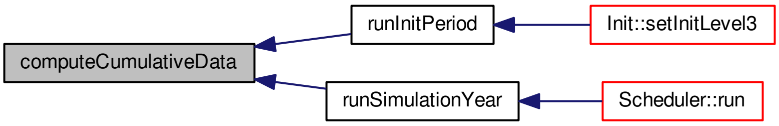
```
01312                                        {
01313 
01314   msgOut(MSG_INFO, "Starting computing some cumulative values..");
01315   int thisYear     =  MTHREAD->SCD->getYear();
01316 
01317 //    double sumCumTP=0;
01318 //    double sumVHa = 0;
01319 //    double count = 0;
01320 //    double avg_sumCumTp;
01321 //    double avg_sumVHa;
01322 
01323   for(uint r2= 0; r2<regIds2.size();r2++){
01324     int regId = regIds2[r2];
01325     regPx = MTHREAD->MD->getRegion(regId)->getMyPixels();
01326 
01327     for (uint p=0;p<regPx.size();p++){
01328       Pixel* px = regPx[p];
01329       px->cumTp.clear();
01330       px->cumTp_exp.clear();
01331       px->vHa_exp.clear();
01332       px->vHa.clear();
01333       px->cumAlive.clear();
01334       px->cumAlive_exp.clear();
01335       double expType = px->expType;
01336 
01337       for(uint j=0;j<fTypes.size();j++){
01338         string ft = fTypes[j];
01339 
01340         double tp_multiplier_now       = px->getMultiplier("tp_multiplier",ft,
      DATA_NOW);
01341         double tp_multiplier_t0        = px->getMultiplier("tp_multiplier",ft,
      firstYear);
01342         double mortCoef_multiplier_now = px->getMultiplier("mortCoef_multiplier",ft,
      DATA_NOW);
01343         double mortCoef_multiplier_t0  = px->getMultiplier("mortCoef_multiplier",ft,
      firstYear);
01344         double betaCoef_multiplier_now = px->getMultiplier("betaCoef_multiplier",ft,
      DATA_NOW);
01345         double betaCoef_multiplier_t0  = px->getMultiplier("betaCoef_multiplier",ft,
      firstYear);
01346         double pathMort_now, pathMort_t0;
01347 
01348         // calculating the cumulative time of passage and the (cumulativelly generated) vHa for each
       diameter class (depending on forest owners diam growth expectations)
01349         //loop(u$(ord(u)=1),
01350         //   cumTp(u,i,lambda,essence) = tp_u1(i,essence,lambda);
01351         //);
01352         //loop(u$(ord(u)>1),
01353         //   cumTp(u,i,lambda,essence) = cumTp(u-1,i,lambda,essence)+tp(u-1,i,lambda,essence);
01354         //);
01355         ////ceil(x) DNLP returns the smallest integer number greater than or equal to x
01356         //loop( (u,i,lambda,essence),
01357         //  cumTp(u,i,lambda,essence) =   ceil(cumTp(u,i,lambda,essence));
01358         //);
01359         vector <double> cumTp_temp;     // cumulative time of passage to REACH a diameter class (tp is to
       LEAVE to the next one)
01360         vector <double> vHa_temp;       // volume at hectar by each diameter class [m^3/ha]
01361         vector <double> cumAlive_temp;     // cumulated alive rate to reach a given diameter class
01362         vector <double> cumTp_exp_temp; // expected version of cumTp_temp
01363         vector <double> vHa_exp_temp;   // expected version of vHa_temp
01364         vector <double> cumAlive_exp_temp; // "expected" version of cumMort
01365 
01366         MD->setErrorLevel(MSG_NO_MSG); // as otherwise on 2007 otherwise sfd()
       will complain that is filling multiple years (2006 and 2007)
01367         for (uint u=0; u<dClasses.size(); u++){
01368           string dc = dClasses[u];
01369           double cumTp_u, cumTp_u_exp, cumTp_u_noExp, cumTp_u_fullExp;
01370           double tp, tp_exp, tp_noExp, tp_fullExp;
01371           double vHa_u, vHa_u_exp, vHa_u_noExp, vHa_u_fullExp, beta, beta_exp, beta_noExp, beta_fullExp,
      mort, mort_exp, mort_noExp, mort_fullExp;
01372           double cumAlive_u, cumAlive_exp_u;
01373           pathMort_now = px->getPathMortality(ft,dc,DATA_NOW);
01374           pathMort_t0 = px->getPathMortality(ft,dc,firstYear);
01375           // only cumTp is depending for the expectations, as it is what it is used by owner to calculate
       return of investments.
```

```
          // the tp, beta and mort coefficients instead are the "real" ones as predicted by scientist for that specific time

          if(u==0) {
            // first diameter class.. expected  and real values are the same (0)
            cumTp_u = 0.;
            vHa_u   = 0.;
            cumAlive_u = 1.;
            cumTp_temp.push_back(cumTp_u);
            vHa_temp.push_back(vHa_u);
            cumTp_exp_temp.push_back(cumTp_u);
            vHa_exp_temp.push_back(vHa_u);
            cumAlive_temp.push_back(cumAlive_u);
            cumAlive_exp_temp.push_back(cumAlive_u);
          } else {
            // other diameter classes.. first dealing with real values and then with expected ones..
            // real values..
            // real values..
            tp = gfd("tp",regId,ft,dClasses[u-1],thisYear)*tp_multiplier_now;
            cumTp_u = cumTp_temp[u-1] + tp;
            if (u==1){
              /**
              Note on the effect of mortality modifiers on the entryVolHa.
              Unfortunaly for how it is defined the mortality multiplier (the ratio with the new mortality rate over the old one) we can't
              compute a entryVolHa based on it. It is NOT infact just like: vHa_adjusted = vHa_orig / mort_multiplier.
              The effect of mortality on the vHa of the first diameter class is unknow, and so we can't compute the effect of a relative
              increase.
              */
              vHa_u = gfd("entryVolHa",regId,ft,"",thisYear);
              mort = 0.; // not info about mortality first diameter class ("00")
            } else {
              mort = 1-pow(1-gfd("mortCoef",regId,ft,dClasses[u-1],thisYear)*mortCoef_multiplier_now+pathMort_now,tp); // mortality of the previous diameter class
              beta = gfd("betaCoef",regId,ft,dc, thisYear)*betaCoef_multiplier_now;
              vHa_u = vHa_temp[u-1]*beta*(1-mort);
            }
            cumAlive_u = max(0.,cumAlive_temp[u-1]*(1-mort));
            cumAlive_temp.push_back(cumAlive_u);
            cumTp_temp.push_back(cumTp_u);
            vHa_temp.push_back(vHa_u);
            // expected values..
            /**
            param expType Specify how the forest owners (those that make the investments) behave will be the time of passage in the future in order to calculate the cumulative time of passage in turn used to discount future revenues.
            Will forest owners behave adaptively believing the time of passage between diameter classes will be like the observed one at time they make decision (0) or they will have full expectations believing forecasts (1) or something in the middle ?
            For compatibility with the GAMS code, a -1 value means using initial simulation tp values (fixed cumTp)."
            */
            if (expType == -1){
              tp_exp = gfd("tp",regId,ft,dClasses[u-1],firstYear)*tp_multiplier_t0;
              //tp = px->tp.at(u); no. not possible, tp stored at pixel level is the current year one
              cumTp_u_exp = cumTp_exp_temp[u-1]+tp_exp;
              cumTp_exp_temp.push_back(cumTp_u_exp);
              if(u==1) {
                vHa_u_exp = gfd("entryVolHa",regId,ft,"",firstYear);
                mort_exp = 0.; // not info about mortality first diameter class ("00")
              } else {
                mort_exp = 1-pow(1-gfd("mortCoef",regId,ft,dClasses[u-1],firstYear)*mortCoef_multiplier_t0+pathMort_t0,tp_exp); // mortality rate of previous diameter class
                beta_exp = gfd("betaCoef",regId,ft,dc, firstYear)*betaCoef_multiplier_t0;
                vHa_u_exp = vHa_exp_temp[u-1]*beta_exp*(1-mort_exp);
              }
            } else {
              double tp_multiplier_dynamic = px->getMultiplier("tp_multiplier",ft,thisYear+ceil(cumTp_exp_temp[u-1]));
              tp_noExp = gfd("tp",regId,ft,dClasses[u-1])*tp_multiplier_now;
              cumTp_u_noExp = cumTp_exp_temp[u-1]+tp_noExp;
              tp_fullExp = gfd("tp",regId,ft,dClasses[u-1],thisYear+ceil(cumTp_exp_temp[u-1]))*tp_multiplier_dynamic ; // time of passage that there should be to reach this diameter class in the year where the previous diameter class will be reached
              cumTp_u_fullExp = cumTp_exp_temp[u-1]+tp_fullExp ; // it adds to the time of passage to reach the previous diameter class the time of passage that there should be to reach this diameter class in the year where the previous diameter class will be reached
              cumTp_u_exp = cumTp_u_fullExp*expType+cumTp_u_noExp*(1-expType); // 20121108: it's math the same as cumTp_exp_temp[u-1] + tp
              cumTp_exp_temp.push_back(cumTp_u_exp);
              if(u==1) {
                vHa_u_noExp   = gfd("entryVolHa",regId,ft,"",DATA_NOW);
                vHa_u_fullExp = gfd("entryVolHa",regId,ft,"",thisYear+ceil(cumTp_u));
                vHa_u_exp = vHa_u_fullExp*expType+vHa_u_noExp*(1-expType);
                mort_exp = 0.; // not info about mortality first diameter class ("00")
              } else {
                mort_noExp = 1-pow(1-min(1.0,gfd("mortCoef",regId,ft,dClasses[u-1],DATA_NOW)*mortCoef_multiplier_now+pathMort_now), tp_noExp); // mortCoef is a yearly value. Mort coeff between class is 1-(1-mortCoeff)^tp
                double mortCoef_multiplier_dynamic = px->getMultiplier("mortCoef_multiplier",ft,thisYear+ceil(cumTp_exp_temp[u-1]));
                double pathMort_dynamic = px->getPathMortality(ft,dc,thisYear+ceil(cumTp_exp_temp[u-1]));
                mort_fullExp = 1-pow(1-min(1.0,gfd("mortCoef",regId,ft,dClasses[u-1],thisYear+ceil(cumTp_exp_temp[u-1]))*mortCoef_multiplier_dynamic+pathMort_dynamic),tp_fullExp); // mortality of the previous diameter class
                //double debug1 = gfd("mortCoef",regId,ft,dClasses[u-1],thisYear+ceil(cumTp_exp_temp[u-1]));
                //double debug2 = debug1*mortCoef_multiplier_dynamic+pathMort_dynamic;
                //double debug3 = min(1.0,debug2);
                //double debug4 = 1.0-debug3;
                //double debug5 = pow(debug4,tp_fullExp);
                //double debug6 = 1.0-debug5;


                beta_noExp   = gfd("betaCoef",regId,ft,dc, DATA_NOW)*betaCoef_multiplier_now;
                double betaCoef_multiplier_dynamic = px->getMultiplier("betaCoef_multiplier",ft,thisYear+ceil(cumTp_u));
                beta_fullExp = gfd("betaCoef",regId,ft,dc, thisYear+ceil(cumTp_u))*betaCoef_multiplier_dynamic;
                mort_exp = mort_fullExp*expType+mort_noExp*(1-expType);
                beta_exp = beta_fullExp*expType+beta_noExp*(1-expType);
                vHa_u_exp = vHa_exp_temp[u-1]*beta_exp*(1-mort_exp); // BUG !!! mort is yearly value, not between diameter class. SOLVED 20121108
              }
            }
            vHa_exp_temp.push_back(vHa_u_exp);
            cumAlive_exp_u = max(0.,cumAlive_exp_temp[u-1]*(1-mort_exp));
            cumAlive_exp_temp.push_back(cumAlive_exp_u);

            //cout << "********" << endl;
            //cout << "dc;mort;cumAlive;cumAlive_exp "<< endl ;
            //cout << dClasses[u] << ";"<< mort << ";" << cumAlive_u << ";" << cumAlive_exp_u << endl;

          }
          // debug stuff on vHa
          //double vHa_new = gfd("vHa",regId,ft,dc,DATA_NOW);
          //double hv2fa_old = gfd("hv2fa",regId,ft,dc,DATA_NOW);
          //cout << "Reg|Ft|dc|vHa (new)|1/hv2fa (old):   " << regId << " | " << ft;
          //cout << " | " << dc << " | " << vHa_new << " | " << 1/hv2fa_old  << endl;

        } // end of each diam
        //double pixID = px->getID();
        //cout << thisYear << ";"<< regIds2[r2] << ";" << pixID << ";" << ft << ";" << cumTp_exp_temp[3] << ";" << vHa_exp_temp[3] << endl;
        px->cumTp.push_back(cumTp_temp);
        px->vHa.push_back(vHa_temp);
        px->cumAlive.push_back(cumAlive_temp);
        px->cumTp_exp.push_back(cumTp_exp_temp);
        px->vHa_exp.push_back(vHa_exp_temp);
        px->cumAlive_exp.push_back(cumAlive_exp_temp);

        //sumCumTP += cumTp_exp_temp[3];
        //sumVHa += vHa_exp_temp[3];
        //count ++;


      } // end of each ft
      double debug = 0.0;
    } // end of each pixel
  } // end of each region
  MD->setErrorLevel(MSG_ERROR);
    //avg_sumCumTp = sumCumTP/ count;
    //avg_sumVHa = sumVHa / count;
    //cout << "Avg sumCumTp_35 and sumVha_35: " << avg_sumCumTp << " and " << avg_sumVHa << " (" << count << ")" << endl;
    //exit(0);
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.26.3.7 double computeExpectedPrice (const double & *curLocPrice,* const double & *worldCurPrice,* const double & *worldFutPrice,* const double & *sl,* const double & *sa,* const double & *expCoef*)

Compute weighted expected price for a given product.

Compute the expectation weighted price based on the ratio of the international (world) price between the future and now.

Parameters

curLocPrice	The local current price
worldCurPrice	The world current price
worldFutPrice	The world future price
sl	Supply local
sa	Supply abroad
expCoef	The expectation coefficient for prices for the agent [0,1]

Returns

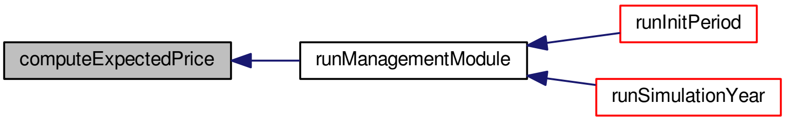
The expType-averaged local (or weighter) price

Definition at line 2017 of file ModelCoreSpatial.cpp.

Referenced by runManagementModule().

```
02017
                                                                                     {
02018   double fullExpWPrice = (curLocPrice*(worldFutPrice/worldCurPrice)*sl+worldFutPrice*sa)/(sa+sl);
02019   double curWPrice = (curLocPrice*sl+worldCurPrice*sa)/(sl+sa);
02020   return curWPrice * (1-expCoef) + fullExpWPrice * expCoef;
02021 }
```

Here is the caller graph for this function:



4.26.3.8 void computeInventary ( )

in=f(vol_t-1)

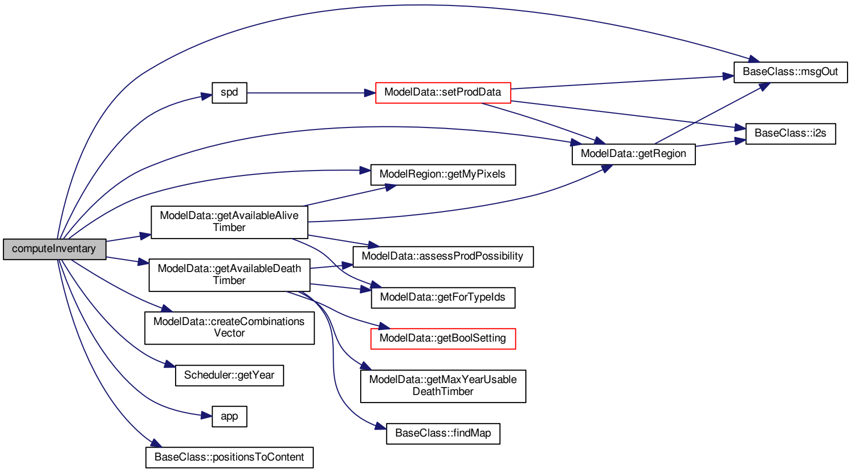
Definition at line 1598 of file ModelCoreSpatial.cpp.

Referenced by runInitPeriod(), and runSimulationYear().

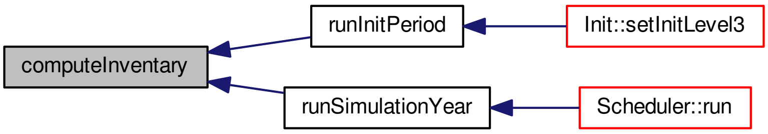
```
01598                                  { // in=f(vol_t-1)
01599   msgOut(MSG_INFO, "Starting computing inventary available for this year..");
01600   int nbounds = pow(2,priProducts.size());
01601   vector<vector<int>> concernedPriProductsTotal = MTHREAD->MD->
      createCombinationsVector(priProducts.size());
01602   int currentYear = MTHREAD->SCD->getYear();
01603
01604   for(uint i=0;i<regIds2.size();i++){
01605     int r2 = regIds2[i];
01606     ModelRegion* REG = MTHREAD->MD->getRegion(r2);
01607     //Gis* GIS = MTHREAD->GIS;
01608     regPx = REG->getMyPixels();
01609     vector <double> in_reg(priProducts.size(),0.);             // should have ceated a vector of
       size priProducts.size(), all filled with zeros
01610     vector <double> in_deathTimber_reg(priProducts.size(),0.); // should have ceated a vector of
       size priProducts.size(), all filled with zeros
01611     for (uint p=0;p<regPx.size();p++){
01612       Pixel* px = regPx[p];
01613       //int debugPx = px->getID();
01614       //int debug2 = debugPx;
01615       //px->in.clear();
01616       for(uint pp=0;pp<priProducts.size();pp++){
01617         double in = 0;
01618         for(uint ft=0;ft<fTypes.size();ft++){
01619           for(uint dc=0;dc<dClasses.size();dc++){
01620             in += app(priProducts[pp],fTypes[ft],dClasses[dc])*px->
      vol_l.at(ft).at(dc)*px->avalCoef;
01621           }
01622         }
01623         //px->in.push_back(in);
01624         in_reg.at(pp) += in;
01625       } // end of each priProduct
```

```
    } // end each pixel


    for(uint pp=0;pp<priProducts.size();pp++){
      vector<string> priProducts_vector;
      priProducts_vector.push_back(priProducts[pp]);

      double in_deathMortality = MD->getAvailableDeathTimber(priProducts_vector,r2,currentYear-1);
      in_deathTimber_reg.at(pp) += in_deathMortality;

      // Even if I fixed all the lower bounds to zero in Opt::get_bounds_info still the model
      // doesn't solve with no-forest in a region.
      // Even with 0.0001 doesn't solve !!
      // With 0.001 some scenarios doesn't solve in 2093
      // With 0.003 vRegFixed doesn't solve in 2096
      // Tried with 0.2 but no changes, so put it back on 0.003
      //spd(max(0.001,in_reg.at(pp)),"in",r2,priProducts[pp],DATA_NOW,true);
      spd(in_reg.at(pp),"in",r2,priProducts[pp],DATA_NOW,true);
      spd(in_deathTimber_reg.at(pp),"in_deathTimber",r2,priProducts[pp],DATA_NOW,true);
      #ifdef QT_DEBUG
      if (in_reg.at(pp) < -0.0){
        msgOut(MSG_CRITICAL_ERROR,"Negative inventory");
      }
      #endif
    }

    // ##### Now creating a set of bonds for the optimisation that account of the fact that the same ft,dc can be used for multiple products:

    // 20160928: Solved a big bug: for each combination instead of taking the UNION of the various priProduct inventory sets I was taking the sum
    // Now both the alive and the death timber are made from the union
    // 20150116: As the same (ft,dc) can be used in more than one product knowing -and bounding the supply in the optimisation- each single
    // in(pp) is NOT enought.
    // We need to bound the supply for each possible combination, that is for 2^(number of prim.pr)
    // Here we compute the detailed inventory. TO.DO: Create the pounds in Opt. done
    // 20160209: Rewritten and corrected a bug that was not giving enought inv to multiproduct combinations
    for (uint i=0; i<nbounds; i++){
      vector<int> concernedPriProducts = concernedPriProductsTotal[i];
      vector<string> concernedPriProducts_ids = positionsToContent(priProducts,concernedPriProducts);
      //double debug       = 0.0;
      //for(uint z=0;z<concernedPriProducts.size();z++){
      //  debug      += gpd("in",r2,priProducts[concernedPriProducts[z]]); // to.do: this will need to be rewritten checked!
      //}
      double bound_alive       = MD->getAvailableAliveTimber(concernedPriProducts_ids,r2); // From px->vol_l, as in "in"
      double bound_deathTimber = MD->getAvailableDeathTimber(concernedPriProducts_ids,r2,currentYear-1); // From deathTimberInventory map
      double bound_total       = bound_alive + bound_deathTimber;

      REG->inResByAnyCombination[i]             = bound_total;
      REG->inResByAnyCombination_deathTimber[i] = bound_deathTimber;
    } // end for each bond
  } // end each region
}
```

Here is the call graph for this function:


computeInventary
spd
ModelData::setProdData
BaseClass::msgOut
BaseClass::i2s
ModelData::getRegion
ModelRegion::getMyPixels
ModelData::getAvailableAlive Timber
ModelData::assessProdPossibility
ModelData::getAvailableDeath Timber
ModelData::getForTypeIds
ModelData::createCombinations Vector
ModelData::getBoolSetting
Scheduler::getYear
ModelData::getMaxYearUsable DeathTimber
app
BaseClass::findMap
BaseClass::positionsToContent

Here is the caller graph for this function:



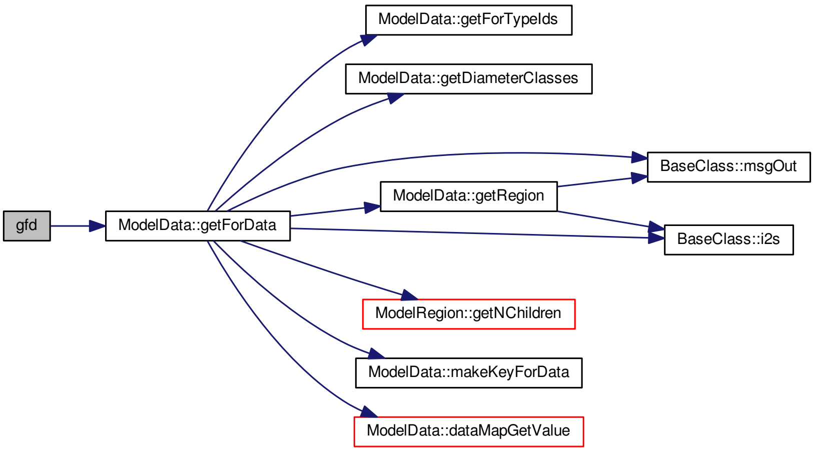
4.26.3.9 double gfd (const string & *type_h*, const int & *regId_h*, const string & *forType_h*, const string & *freeDim_h*, const int & *year* = DATA_NOW) const [inline]

Definition at line 117 of file ModelCoreSpatial.h.

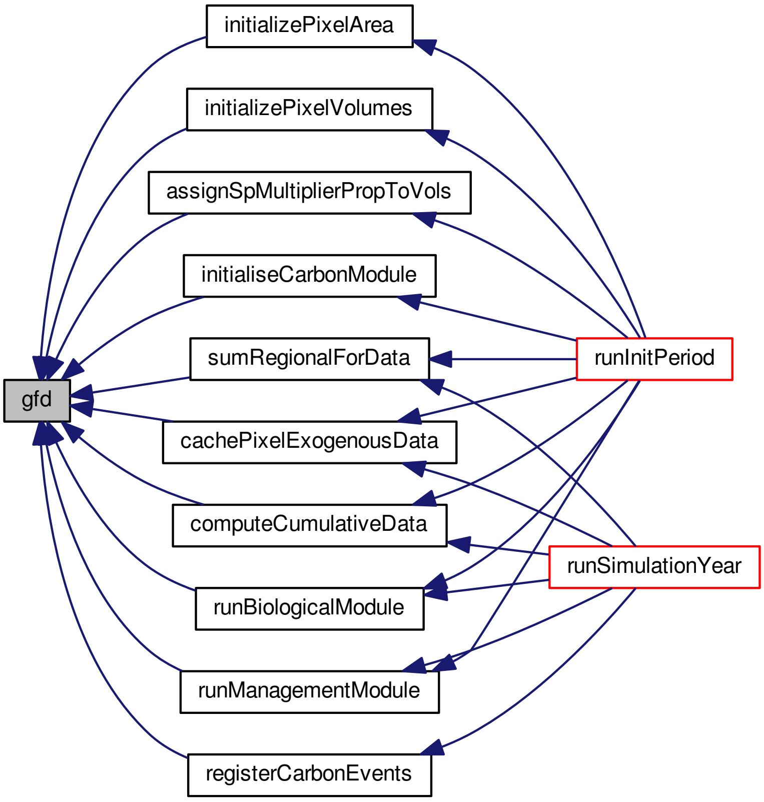
Referenced by assignSpMultiplierPropToVols(), cachePixelExogenousData(), computeCumulativeData(), initialiseCarbonModule(), initializePixelArea(), initializePixelVolumes(), registerCarbonEvents(), runBiologicalModule(), runManagementModule(), and sumRegionalForData().

```
00117 {return MTHREAD->MD->getForData(type_h, regId_h, forType_h, freeDim_h, year);};
```

Here is the call graph for this function:


gfd
ModelData::getForData
ModelData::getForTypeIds
ModelData::getDiameterClasses
BaseClass::msgOut
ModelData::getRegion
BaseClass::i2s
ModelRegion::getNChildren
ModelData::makeKeyForData
ModelData::dataMapGetValue

Here is the caller graph for this function:



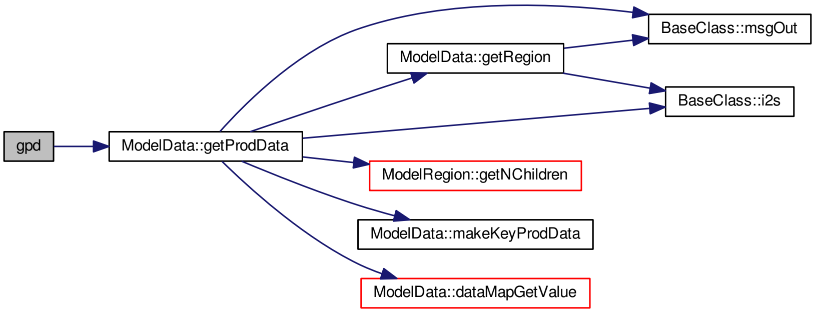
4.26.3.10 double gpd (const string & *type_h,* const int & *regId_h,* const string & *prodId_h,* const int & *year* = DATA_NOW, const string & *freeDim_h* = "") const [inline]

Definition at line 116 of file ModelCoreSpatial.h.

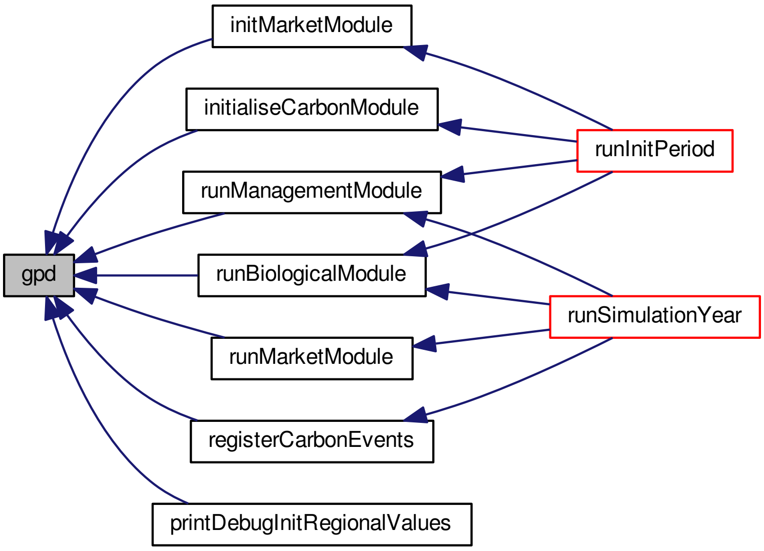
Referenced by initialiseCarbonModule(), initMarketModule(), printDebugInitRegionalValues(), registerCarbonEvents(), runBiologicalModule(), runManagementModule(), and runMarketModule().

```
00116 {return MTHREAD->MD->getProdData(type_h, regId_h, prodId_h, year, freeDim_h);};
```

Here is the call graph for this function:


BaseClass::msgOut
ModelData::getRegion
BaseClass::i2s
gpd
ModelData::getProdData
ModelRegion::getNChildren
ModelData::makeKeyProdData
ModelData::dataMapGetValue

Here is the caller graph for this function:



4.26.3.11 void initialiseCarbonModule ( )

call initialiseDeathBiomassStocks(), initialiseProductsStocks() and initialiseEmissionCounters()

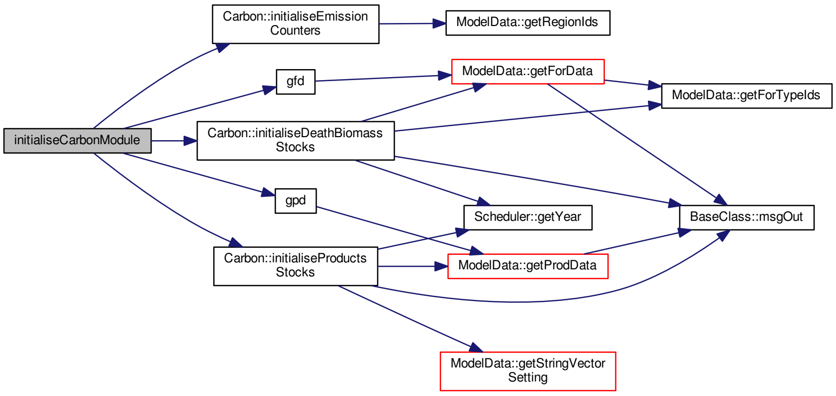
< call initialiseDeathBiomassStocks(), initialiseProductsStocks() and initialiseEmissionCounters()

Definition at line 1188 of file ModelCoreSpatial.cpp.

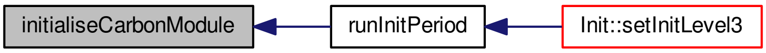
Referenced by runInitPeriod().

```
01188                                         {
01189
01190   ///< call initialiseDeathBiomassStocks(), initialiseProductsStocks() and initialiseEmissionCounters()
01191   MTHREAD->CBAL->initialiseEmissionCounters();
01192
01193   for(uint i=0;i<regIds2.size();i++){
01194     vector<double> deathBiomass;
01195     for(uint j=0;j<fTypes.size();j++){
01196       double deathBiomass_ft = gfd("vMort",regIds2[i],fTypes[j],
      DIAM_ALL,DATA_NOW);
01197       deathBiomass.push_back(deathBiomass_ft);
01198     }
01199     MTHREAD->CBAL->initialiseDeathBiomassStocks(deathBiomass,
      regIds2[i]);
01200     vector<double>qProducts;
01201     for(int p=0;p<priProducts.size();p++){
01202       // for the primary products we consider only the exports as the domestic consumption is entirelly
       transformed in secondary products
01203       double int_exports = gpd("sa",regIds2[i],priProducts[p],
      DATA_NOW);
01204       qProducts.push_back(int_exports);
01205     }
01206     for(int p=0;p<secProducts.size();p++){
01207       // for the tranformed product we skip those that are imported, hence derived from other forest
       systems
01208       double consumption = gpd("dl",regIds2[i],secProducts[p],
      DATA_NOW); // dl = sl + net regional imports
01209       qProducts.push_back(consumption);
01210     }
01211     MTHREAD->CBAL->initialiseProductsStocks(qProducts,
      regIds2[i]);
01212
01213   }
01214 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.26.3.12 void initialiseDeathTimber ( )

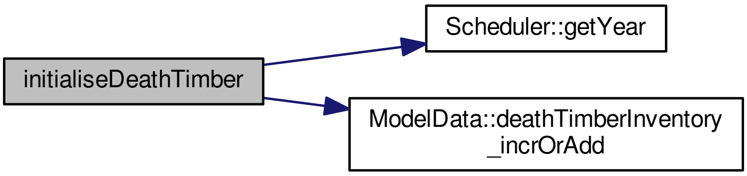
Set deathTimberInventory to zero for the previous years (under the hipotesis that we don't have advanced stock of death biomass usable as timber at the beginning of the simulation)

Definition at line 1217 of file ModelCoreSpatial.cpp.

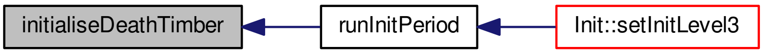
Referenced by runInitPeriod().

```
                                          {
  int currentYear = MTHREAD->SCD->getYear();
  for(int y=currentYear;y>currentYear-30;y--){
    for(uint i=0;i<regIds2.size();i++){
      for(uint j=0;j<fTypes.size();j++){
        for (uint u=0;u<dClasses.size();u++){
          iisskey key(y,regIds2[i],fTypes[j],dClasses[u]);
          MD->deathTimberInventory_incrOrAdd(key,0.0);
        }
      }
    }
  }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.26.3.13 void initializePixelArea ( )

compute px->area for each ft and dc

ModelCoreSpatial::initializePixelArea.

This function compute the initial area by ft and dc. It requires vHa computed in computeCumulativeData, this is why it is separated form the other initialisedPixelValues(). As the sum of area computed using vHa may differ from the one memorised in forArea_∗ layer, all values are scaled to match it before being memorised. Also assign area = area_l

Todo here I have finally also area_ft_dc_px and I can implement the new one I am in 2006

Todo : also update area_l

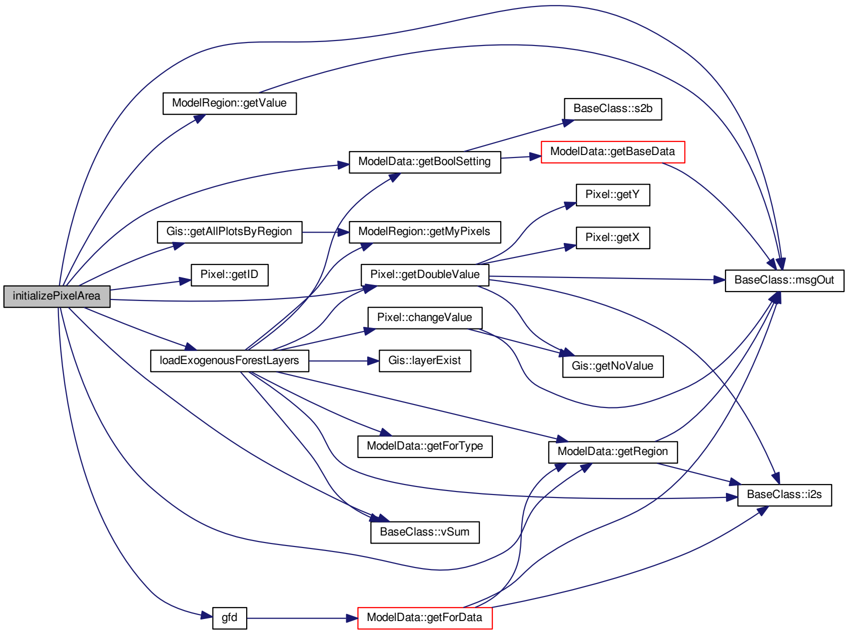
Definition at line 1242 of file ModelCoreSpatial.cpp.

Referenced by runInitPeriod().

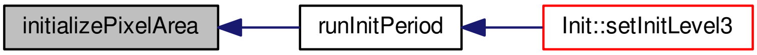
```
01242                                      {
01243   msgOut(MSG_INFO, "Starting initializing pixel-level area");
01244   if(!MD->getBoolSetting("usePixelData")) return;
01245   for(uint i=0;i<regIds2.size();i++){
01246     ModelRegion* reg = MD->getRegion(regIds2[i]);
01247     vector <Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(
      regIds2[i]);
01248     for (uint p=0;p<rpx.size();p++){
01249       Pixel* px = rpx[p];
01250       double pxid= px->getID();
01251       for(uint j=0;j<fTypes.size();j++){
01252         string ft = fTypes[j];
01253         vector <double> tempAreas;
01254         vector <double> areasByFt;
01255         double pxArea = px->getDoubleValue("forArea_"+ft,true)/10000.0; //ha
01256         for (uint u=0;u<dClasses.size();u++){
01257           if(u==0){
01258             double regionArea = reg->getValue("forArea_"+ft,OP_SUM)/10000.0; //ha
01259             double regRegVolumes = gfd("vReg",regIds2[i],ft,""); // regional regeneration
       volumes.. ugly name !!
01260             double newVReg = regionArea ? regRegVolumes*pxArea/regionArea : 0.0;
01261             double tp_u0 = px->tp.at(j).at(0); // time of passage to reach the first production diameter
       class
01262             double entryVolHa = gfd("entryVolHa",regIds2[i],ft,"");
01263             double tempArea = (newVReg*1000000.0/entryVolHa)*tp_u0;
01264             tempAreas.push_back(tempArea);
01265           } else {
01266             string dc = dClasses[u];
01267             double dcVol = px->vol_l.at(j).at(u)*1000000.0; // m^3
01268             double dcVHa = px->vHa.at(j).at(u); // m^3/ha
01269             #ifdef QT_DEBUG
01270             if(dcVol < 0.0 || dcVHa < 0.0){
01271                 msgOut(MSG_CRITICAL_ERROR, "Negative volumes or density in
       initializePixelArea");
01272             }
01273             #endif
01274             double tempArea = dcVHa?dcVol/dcVHa:0;
01275             tempAreas.push_back(tempArea);
01276           }
01277 
01278         } // end dc
01279         double sumTempArea = vSum(tempAreas);
01280        // double sharedc0 = 5.0/90.0; // an arbitrary share of total area allocated to first diameter class
01281         //tempAreas.at(0) = sumTempArea * sharedc0;
01282         //sumTempArea = vSum(tempAreas);
01283         double normCoef = sumTempArea?pxArea/ sumTempArea:0;
01284         //cout << i << ’\t’ << pxid << ’\t’ << ft << ’\t’ << normCoef << endl;
01285         #ifdef QT_DEBUG
01286         if(normCoef < 0.0){
01287             msgOut(MSG_CRITICAL_ERROR, "Negative normCoef in initializePixelArea");
01288         }
01289         #endif
01290         for (uint u=0;u<dClasses.size();u++){
01291           areasByFt.push_back(tempAreas.at(u)*normCoef); //manca la costruzione originale del vettore
01292         }
```

```
        #ifdef QT_DEBUG
        if (pxArea != 0.0){
            double ratio = vSum(areasByFt)/ pxArea;  // vSum(areasByFt) should be equal to pxArea
            if(ratio < 0.99999999999 || ratio > 1.00000000001) {
              msgOut(MSG_CRITICAL_ERROR, "pxArea is not equal to vSum(areasByFt) in
 initializePixelArea");
            }
        }
        #endif
        px->area_l.push_back(areasByFt);
        /// \todo here I have finally also area_ft_dc_px and I can implement the new one I am in 2006
      } // end ft
      px->area = px->area_l;  //Assigning initial value of area to the area of the old year
    } // end px
  } // end region
  loadExogenousForestLayers("area");
  /// \todo: also update area_l
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.26.3.14 void initializePixelVolumes ( )

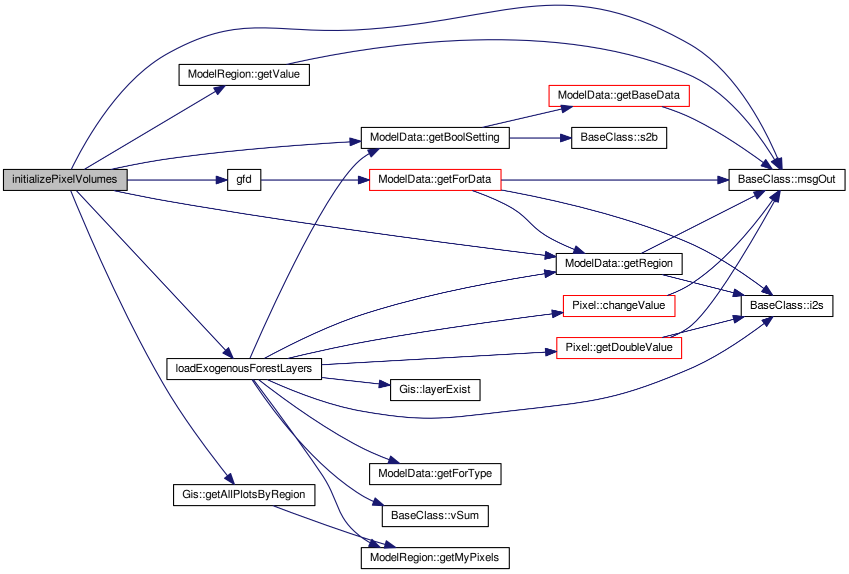
distribuite regional exogenous volumes to pixel volumes using corine land cover area as weight

Definition at line 1050 of file ModelCoreSpatial.cpp.

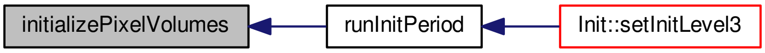
Referenced by runInitPeriod().

```
                                        {
  msgOut(MSG_INFO, "Starting initializing pixel-level values");

  // pxVol = regVol * pxArea/regForArea
  // this function can be done only at the beginning of the model, as it assume that the distribution of volumes by diameter class in the pixels within a certain region is homogeneous, but as the model progress along the time dimension this is no longer true.
  if(!MD->getBoolSetting("usePixelData")) return;
  for(uint i=0;i<regIds2.size();i++){
    ModelRegion* reg = MD->getRegion(regIds2[i]);
    vector <Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(regIds2[i]);
    for (uint j=0;j<rpx.size();j++){
      int debugPx = rpx[j]->getID();
      int debug2 = debugPx;
      rpx[j]->vol.clear(); // not actually necessary
      for(uint y=0;y<fTypes.size();y++){
        vector <double> vol_byu;
        double regForArea = reg->getValue("forArea_"+fTypes[y]);
        for (uint z=0;z<dClasses.size();z++){
          double regVol;
          regVol = z ? gfd("vol",regIds2[i],fTypes[y],dClasses[z],firstYear) : 0 ; // if z=0-> regVol= gfd(), otherwise regVol=0;
          double pxArea = rpx[j]->getDoubleValue("forArea_"+fTypes[y], true); // bug solved 20121109. get zero for not data
          if (pxArea<0.0){
            msgOut(MSG_CRITICAL_ERROR,"Error in initializePixelVolumes, negative pxArea!");
          }
          double pxVol = regForArea ? regVol * pxArea/regForArea: 0; // if we introduce new forest types without initial area we must avoid a 0/0 division
          //rpx[j]->changeValue(pxVol,"vol",fTypes[y],dClasses[z],firstYear);
          vol_byu.push_back(pxVol);
        }
        rpx[j]->vol.push_back(vol_byu);
      }
    }
  }
  loadExogenousForestLayers("vol");
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.26.3.15 void initMarketModule ( )

computes st and pw for second year and several needed-only-at-t0-vars for the market module

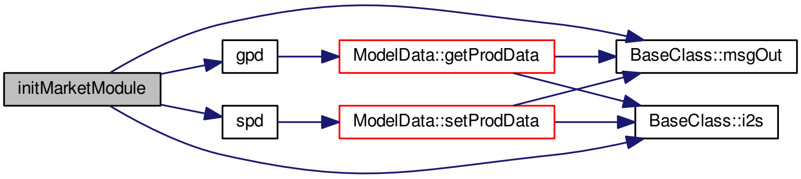
Definition at line 94 of file ModelCoreSpatial.cpp.

Referenced by runInitPeriod().

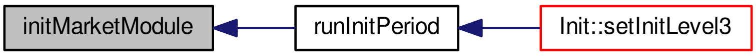
```
00094                                    {
00095   msgOut(MSG_INFO, "Starting market module (init stage)..");
00096
00097   for(uint i=0;i<regIds2.size();i++){
00098     int r2 = regIds2[i];
00099     //RPAR('pl',i,p_tr,t-1)   =  sum(p_pr, a(p_pr,p_tr)*RPAR('pl',i,p_pr,t-1))+m(i,p_tr);
00100     for(uint sp=0;sp<secProducts.size();sp++){
00101       double value = 0;
00102       for (uint pp=0;pp<priProducts.size();pp++){
00103         value += gpd("pl",r2,priProducts[pp],secondYear)*
00104         gpd("a",r2,priProducts[pp],secondYear,
      secProducts[sp]);
00105       }
00106       value += gpd("m",r2,secProducts[sp],secondYear);
00107       spd(value,"pl",r2,secProducts[sp],secondYear,true);
00108     }
00109     // RPAR('dl',i,p_pr,t-1)   =  sum(p_tr, a(p_pr,p_tr)*RPAR('sl',i,p_tr,t-1));
00110     for (uint pp=0;pp<priProducts.size();pp++){
00111       double value=0;
00112       for(uint sp=0;sp<secProducts.size();sp++){
00113         value += gpd("sl",r2,secProducts[sp],secondYear)*
00114         gpd("a",r2,priProducts[pp],secondYear,
      secProducts[sp]);
00115       }
00116       spd(value,"dl",r2,priProducts[pp],secondYear,true);
00117     }
00118     // RPAR('st',i,prd,t-1)    =  RPAR('sl',i,prd,t-1)+RPAR('sa',i,prd,t-1);
00119     // RPAR('dt',i,prd,t-1)    =  RPAR('dl',i,prd,t-1)+RPAR('da',i,prd,t-1);
00120     for (uint ap=0;ap<allProducts.size();ap++){
00121       //double debug = gpd("dl",r2,allProducts[ap],secondYear);
00122       double stvalue =   gpd("sl",r2,allProducts[ap],secondYear)
00123                        + gpd("sa",r2,allProducts[ap],secondYear);
00124       double dtvalue =   gpd("dl",r2,allProducts[ap],secondYear)
00125                        + gpd("da",r2,allProducts[ap],secondYear);
00126       spd(stvalue,"st",r2,allProducts[ap],secondYear,true);
00127       spd(stvalue,"stFromHarvesting",r2,allProducts[ap],secondYear,true);
00128       spd(dtvalue,"dt",r2,allProducts[ap],secondYear,true);
00129     }
00130
00131     // q1(i,p_tr)  =
       1/(1+((RPAR('dl',i,p_tr,t-1)/RPAR('da',i,p_tr,t-1))**(1/psi(i,p_tr)))*(RPAR('pl',i,p_tr,t-1)/PT(p_tr,t-1)));
00132     // p1(i,p_tr)                =  1-q1(i,p_tr);
00133     // RPAR('dc',i,p_tr,t-1)   =  (q1(i,p_tr)*RPAR('da',i,p_tr,t-1)**((psi(i,p_tr)-1)/psi(i,p_tr))+
       p1(i,p_tr)*RPAR('dl',i,p_tr,t-1)**((psi(i,p_tr)-1)/psi(i,p_tr)))**(psi(i,p_tr)/(psi(i,p_tr)-1));
00134     // RPAR('pc',i,p_tr,t-1)   =
       (RPAR('da',i,p_tr,t-1)/RPAR('dc',i,p_tr,t-1))*PT(p_tr,t-1)+(RPAR('dl',i,p_tr,t-1)/RPAR('dc',i,p_tr,t-1))*RPAR('pl',i,p_
00135     // RPAR('pc',i,p_pr,t-1)   =
       (RPAR('sa',i,p_pr,t-1)/RPAR('sc',i,p_pr,t-1))*PT(p_pr,t-1)+(RPAR('sl',i,p_pr,t-1)/RPAR('sc',i,p_pr,t-1))*RPAR('pl',i,p_
00136     // RPAR('pw',i,p_tr,t-1)   =
       (RPAR('dl',i,p_tr,t-1)*RPAR('pl',i,p_tr,t-1)+RPAR('da',i,p_tr,t-1)*PT(p_tr,t-1))/RPAR('dt',i,p_tr,t-1) ; //changed 201
00137     // K(i,p_tr,t-1)             =  k1(i,p_tr)*RPAR('sl',i,p_tr,t-1);
00138     for(uint sp=0;sp<secProducts.size();sp++){
00139       double psi = gpd("psi",r2,secProducts[sp],secondYear);
00140       double dl  = gpd("dl",r2,secProducts[sp],secondYear);
00141       double da  = gpd("da",r2,secProducts[sp],secondYear);
00142       double pl  = gpd("pl",r2,secProducts[sp],secondYear);
```

```
      double sl  = gpd("sl",r2,secProducts[sp],secondYear);
      double k1  = gpd("k1",r2,secProducts[sp],secondYear);
      double pWo = gpd("pl",WL2,secProducts[sp],secondYear); // World price (local price for region 99999)


      double q1  = 1/ ( 1+pow(dl/da,1/psi)*(pl/pWo) );
      double p1  = 1-q1;
      double dc  = pow(
              q1*pow(da,(psi-1)/psi) + p1*pow(dl,(psi-1)/psi)
             ,
              psi/(psi-1)
             );
      double pc = (da/dc)*pWo
             +(dl/dc)*pl;
      double pw = (dl*pl+da*pWo)/(dl+da);
      double k = k1*sl;

      spd(q1,"q1",r2,secProducts[sp],firstYear,true);
      spd(p1,"p1",r2,secProducts[sp],firstYear,true);
      spd(dc,"dc",r2,secProducts[sp],secondYear,true);
      spd(pc,"pc",r2,secProducts[sp],secondYear,true);
      spd(pw,"pw",r2,secProducts[sp],secondYear,true);
      spd(k,"k",r2,secProducts[sp],secondYear,true);
    }

    // t1(i,p_pr)               = 1/(1+((RPAR(’sl’,i,p_pr,t-1)/RPAR(’sa’,i,p_pr,t-1))**(1/eta(i,p_pr)))*(RPAR(’pl’,i,p_pr,t-1)/PT(p_pr,t-1)));
    // r1(i,p_pr)               =  1-t1(i,p_pr);
    // RPAR(’sc’,i,p_pr,t-1)    =  (t1(i,p_pr)*RPAR(’sa’,i,p_pr,t-1)**((eta(i,p_pr)-1)/eta(i,p_pr))+ r1(i,p_pr)*RPAR(’sl’,i,p_pr,t-1)**((eta(i,p_pr)-1)/eta(i,p_pr)))**(eta(i,p_pr)/(eta(i,p_pr)-1))
    // RPAR(’pc’,i,p_pr,t-1)    = (RPAR(’sa’,i,p_pr,t-1)/RPAR(’sc’,i,p_pr,t-1))*PT(p_pr,t-1)+(RPAR(’sl’,i,p_pr,t-1)/RPAR(’sc’,i,p_pr,t-1))*RPAR(’pl’,i,p_
    // RPAR(’pw’,i,p_pr,t-1)    = (RPAR(’sl’,i,p_pr,t-1)*RPAR(’pl’,i,p_pr,t-1)+RPAR(’sa’,i,p_pr,t-1)*PT(p_pr,t-1))/RPAR(’st’,i,p_pr,t-1) ; //changed 201
    for(uint pp=0;pp<priProducts.size();pp++){

      double sl  = gpd("sl",r2,priProducts[pp],secondYear);
      double sa  = gpd("sa",r2,priProducts[pp],secondYear);
      double eta = gpd("eta",r2,priProducts[pp],secondYear);
      double pl  = gpd("pl",r2,priProducts[pp],secondYear);
      double pWo = gpd("pl",WL2,priProducts[pp],secondYear); // World price (local price for region 99999)


      double t1 = 1/ ( 1+(pow(sl/sa,1/eta))*(pl/pWo) );
      double r1 = 1-t1;
      double sc = pow(
               t1*pow(sa,(eta-1)/eta) + r1*pow(sl,(eta-1)/eta)
            ,
               eta/(eta-1)
            );
      double pc = (sa/sc)*pWo+(sl/sc)*pl;
      double pw = (sl*pl+sa*pWo)/(sl+sa);

      spd(t1,"t1",r2,priProducts[pp],firstYear,true);
      spd(r1,"r1",r2,priProducts[pp],firstYear,true);
      spd(sc,"sc",r2,priProducts[pp],secondYear,true);
      spd(pc,"pc",r2,priProducts[pp],secondYear,true);
      spd(pw,"pw",r2,priProducts[pp],secondYear,true);
    }

    // up to here tested with gams output on 20120628, that’s fine !!
  } // end for each region in level 2


  // initializing the exports to zero quantities
  // initializing  ofthe transport cost for the same region to one and distance to zero
  for(uint r1=0;r1<l2r.size();r1++){
    for(uint r2=0;r2<l2r[r1].size();r2++){
      for(uint p=0;p<allProducts.size();p++){
        for(uint r2To=0;r2To<l2r[r1].size();r2To++){
          spd(0,"rt",l2r[r1][r2],allProducts[p],secondYear,true, i2s(l2r[r1][r2To]));  // regional trade, it was exp in gams
          if(l2r[r1][r2] == l2r[r1][r2To]){
            spd(1,"ct",l2r[r1][r2],allProducts[p],firstYear,true, i2s(l2r[r1][r2To])); // as long this value is higher than zero, rt within the same region is not choosen by the solver, so the value doesn’t really matters. If it is zero, the solver still works and results are the same, but reported rt within the region are crazy high (100000)
          }
        }
      } // end each product

      for(uint r2To=0;r2To<l2r[r1].size();r2To++){
        if(l2r[r1][r2] == l2r[r1][r2To]){
          spd(0,"dist",l2r[r1][r2],"",firstYear,true,i2s(l2r[r1][r2To])); // setting distance zero in code, so no need to put it in the data
        }
      }
    } // end of r2 regions
  } // end of r1 region
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.26.3.16 void loadExogenousForestLayers (const string & *what*)

Set pixel volumes (what="vol") OR areas (what="area") by specific forest types as defined in gis layers for volumes and proportionally to volumes for areas.

It uses volumes from gis data to "move" volumes from one forest type to the other (when called with what="vol"). Then it moves areas proportionally and, as dc0 volumes are not defined but area it is, compute, again proportionally, area in destination forest times for dc=0 It acts on the pix->vol, pix->area and pix->area_l vectors. It also create/update the px->values layer map for the area, but it doesn't cash the results in forDataMap.

It is called first with parameter what="vol" in initializePixelVolumes() and then with what="area" in initializePixelAreas(). As we need the original volumes in the area allocation, original_vols is set as a static variable. Allocate area proportionally to volumes (see file test_proportional_computation_of_areas_from_volumes.ods) Example: FtIn FtOut Vtrasfer con ash 0.2 brHf ash 0.1 brCopp ash 0.3 con oak 0.3 brHf oak 0.2 brCopp oak 0.1

```
   Vorig Aorig Vnew  Anew
```

con 10 30 9.5 28.5 Aorig-Aorig∗(Vtrasfer1/Vorig)-Aorig(Vtrasfer2/Vorig) brHf 5 20 4.7 18.8 brCopp 2 20 1.6 16 ash 0 0 0.6 4 Aorig1∗Vtrasfer1/(Vorig1)+Aorig2∗Vtrasfer2/(Vorig2)+... oak 0 0 0.6 2.7 70 70

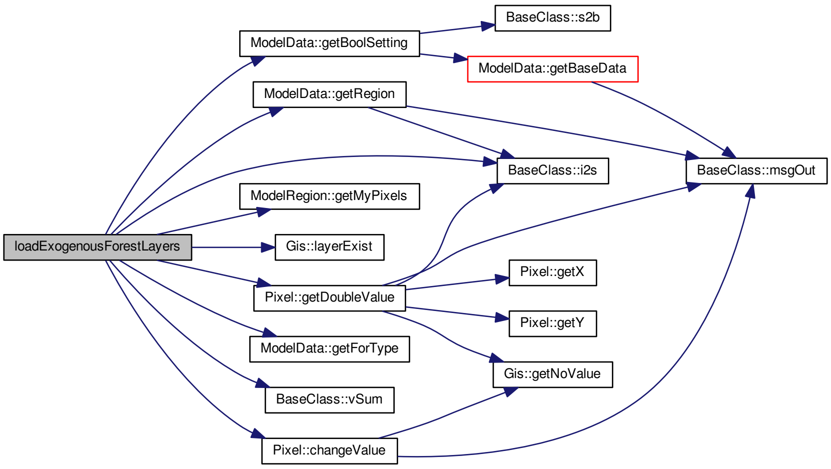
Definition at line 2034 of file ModelCoreSpatial.cpp.

Referenced by initializePixelArea(), and initializePixelVolumes().

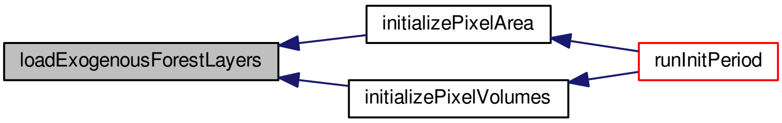
```
                                                               {
  if(!MD->getBoolSetting("useSpExplicitForestTypes")) return;

  int nFTypes = fTypes.size();
  int nDC     = dClasses.size();
  int pxC     = 0;

  for(uint ir=0;ir<regIds2.size();ir++){
    int r2 = regIds2[ir];
    ModelRegion* REG = MTHREAD->MD->getRegion(r2);
    regPx = REG->getMyPixels();
    pxC += regPx.size();
  }

  static vector<vector<vector<double>>> original_vols(pxC, vector<vector<double>>(nFTypes, vector<double>(nDC, 0.0))); // by px counter, ftype, dc

  if(what=="vol"){
    // first, before transfering volumes, saving the original ones..
    for(uint i=0;i<fTypes.size();i++){
      for (uint u=0; u<dClasses.size(); u++){
        int pxC_loc = 0;
        for(uint ir=0;ir<regIds2.size();ir++){
          int r2 = regIds2[ir];
          ModelRegion* REG = MTHREAD->MD->getRegion(r2);
          regPx = REG->getMyPixels();
          for (uint p=0;p<regPx.size();p++){
            Pixel* px = regPx[p];
            original_vols[pxC_loc][i][u] += px->vol[i][u];
            pxC_loc ++;
          }
        }
      }
    }
    for(uint i=0;i<fTypes.size();i++){
      string fti = fTypes[i];
      for(uint o=0;o<fTypes.size();o++){
        string fto = fTypes[o];
        for (uint u=1; u<dClasses.size(); u++){ // first diameter class volumes are computed from the model..
          string layerName =  "spInput#vol#"+fto+"#"+fti+"#"+i2s(u);
          if (MTHREAD->GIS->layerExist(layerName)){
            for(uint ir=0;ir<regIds2.size();ir++){
              int r2 = regIds2[ir];
              ModelRegion* REG = MTHREAD->MD->getRegion(r2);
              regPx = REG->getMyPixels();
              for (uint p=0;p<regPx.size();p++){
                Pixel* px = regPx[p];
                double vol_transfer = min(px->getDoubleValue(layerName,true)/1000000,px->vol[i][u]) ; // Vol in the layer are in m^3, in the model in Mm^3
                px->vol[i][u] -= vol_transfer;
                px->vol[o][u] += vol_transfer;
              }
            }
          }
        }
      }
    }
  }

  if(what=="area"){
    /**
    Allocate area proportionally to volumes (see file test_proportional_computation_of_areas_from_volumes.ods)
    Example:
    FtIn   FtOut Vtrasfer
    con    ash   0.2
    brHf   ash   0.1
    brCopp ash   0.3
    con    oak   0.3
    brHf   oak   0.2
    brCopp oak   0.1

           Vorig Aorig Vnew  Anew
    con    10    30    9.5   28.5  Aorig-Aorig*(Vtrasfer1/Vorig)-Aorig(Vtrasfer2/Vorig)
    brHf   5     20    4.7   18.8
    brCopp 2     20    1.6   16
    ash    0     0     0.6   4     Aorig1*Vtrasfer1/(Vorig1)+Aorig2*Vtrasfer2/(Vorig2)+...
    oak    0     0     0.6   2.7
                 70          70
    */
    // first, before transfering areas, saving the original ones (we already saved the vols in the what="vol" section, that is called before this one)..
    vector<vector<vector<double>>> original_areas(pxC, vector<vector<double>>(nFTypes, vector<double>(nDC, 0.0))); // by px counter, ftype, dc
    for(uint i=0;i<fTypes.size();i++){
      for (uint u=0; u<dClasses.size(); u++){
        int pxC_loc = 0;
        for(uint ir=0;ir<regIds2.size();ir++){
          int r2 = regIds2[ir];
          ModelRegion* REG = MTHREAD->MD->getRegion(r2);
          regPx = REG->getMyPixels();
          for (uint p=0;p<regPx.size();p++){
            Pixel* px = regPx[p];
            original_areas[pxC_loc][i][u] += px->area_l[i][u];
            pxC_loc ++;
          }
        }
      }
    }


    // transferred areas ordered by pxcounter, i and then o ftype. Used to then repart the 0 diameter class..
    vector<vector<vector<double>>> transferred_areas(pxC, vector<vector<double>>(nFTypes, vector<double>(nFTypes, 0.0))); // initialize a 3d vector of nFTypes zeros.

    for(uint i=0;i<fTypes.size();i++){
      string fti = fTypes[i];
      for(uint o=0;o<fTypes.size();o++){
        string fto = fTypes[o];
        for (uint u=1; u<dClasses.size(); u++){ // first diameter class area is comuted proportionally..
          string layerName =  "spInput#vol#"+fto+"#"+fti+"#"+i2s(u);
          if (MTHREAD->GIS->layerExist(layerName)){
            int pxC_loc = 0;
            for(uint ir=0;ir<regIds2.size();ir++){
              int r2 = regIds2[ir];
              ModelRegion* REG = MTHREAD->MD->getRegion(r2);
              regPx = REG->getMyPixels();
              for (uint p=0;p<regPx.size();p++){
                Pixel* px = regPx[p];
                double vol_i_orig    = original_vols[pxC_loc][i][u];
                double vol_transfer  = vol_i_orig?px->getDoubleValue(layerName,true)/1000000:0.0; // Vol in the layer are in m^3, in the model in Mm^3
                double area_i_orig   = original_areas[pxC_loc][i][u];
                double area_transfer = vol_i_orig?area_i_orig*vol_transfer/vol_i_orig:0.0;
                px->area_l[i][u] -=  area_transfer;
                px->area[i][u]    = px->area_l[i][u];
                px->area_l[o][u] += area_transfer;
                px->area[o][u]    = px->area_l[o][u];
                transferred_areas[pxC_loc][i][o] += area_transfer;
                pxC_loc ++;
             }
           }
          }
        }
      }
    }

    // Moving the area in the 0 diameter class, for which no info is normally available..
    double pxC_loc = 0;
    for(uint ir=0;ir<regIds2.size();ir++){
      int r2 = regIds2[ir];
      ModelRegion* REG = MTHREAD->MD->getRegion(r2);
      regPx = REG->getMyPixels();
      for (uint p=0;p<regPx.size();p++){
        Pixel* px = regPx[p];
        for(uint i=0;i<fTypes.size();i++){
          for(uint o=0;o<fTypes.size();o++){
            double area_i_orig = 0.0;
            for (uint u=1; u<dClasses.size(); u++){ // we want to skip the 0 diameter class, this is why we don’t simply use vSum()..
              area_i_orig += original_areas[pxC_loc][i][u];
            }
            double area_transfer_u0 = area_i_orig?original_areas[pxC_loc][i][0]*(transferred_areas[pxC_loc][i][o]/area_i_orig):0.0;
            px->area_l[i][0] -= area_transfer_u0 ;
            px->area[i][0] = px->area_l[i][0];
            px->area_l[o][0] += area_transfer_u0 ; // bug corrected 20151130: it was 0 instead of o (output) !!
            px->area[o][0] = px->area_l[0][0];     // bug corrected 20151130: it was 0 instead of o (output) !!
          }
        }
        pxC_loc++;
      }
    }

    // Alligning the area memorised in the px layers to the new areas of the ft..
    for(uint i=0;i<fTypes.size();i++){
      string fti_id = fTypes[i];
      forType* fti  = MTHREAD->MD->getForType(fti_id);
      int ft_memType = fti->memType;
```

```
   string ft_layerName = fti->forLayer;
   //if(ft_memType==3){
   //  MTHREAD->GIS->addLayer(ft_layerName,ft_layerName,false,true); //20151130: no needed as we already added it in applyForestReclassification (yes, odd, as memory type 3 layerrs do not have any reclassification rule associated, but if I don't add the layer at that time I got other errors)
   // }
   if(ft_memType==3 ||ft_memType==2){
    for(uint ir=0;ir<regIds2.size();ir++){
     int r2 = regIds2[ir];
     ModelRegion* REG = MTHREAD->MD->getRegion(r2);
     regPx = REG->getMyPixels();
     for (uint p=0;p<regPx.size();p++){
      Pixel* px = regPx[p];
      double area_px = vSum(px->area[i]);
      px->changeValue(ft_layerName,area_px*10000);
     }
    }
   }
  }
 } // end if what is area
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



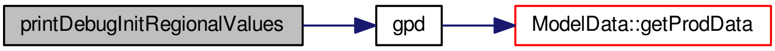
4.26.3.17 void printDebugInitRegionalValues ( )

print initial inv, st, sl and sa in each region

Definition at line 2215 of file ModelCoreSpatial.cpp.

```
02215                                               {
02216   // Print debug stats on inventory and supplies in each region..
02217   cout << "Printing debug information on initial regional inventories and supplies.." << endl;
02218   cout << "Reg\tProduct\t\tInv\tSt\tSa\tSl" << endl;
02219   for(uint r1=0;r1<l2r.size();r1++){
02220     for(uint r2c=0;r2c<l2r[r1].size();r2c++){
02221       for(uint p=0;p<priProducts.size();p++){
02222         int r2 = l2r[r1][r2c];
02223         double inv = gpd("in",r2,priProducts[p],secondYear);
02224         double st = gpd("st",r2,priProducts[p],secondYear);
02225         double sl = gpd("sl",r2,priProducts[p],secondYear);
02226         double sa = gpd("sa",r2,priProducts[p],secondYear);
02227         cout << r2 << "\t" << priProducts[p] << "\t\t" << inv << "\t" << st << "\t" << sl << "\t
      " << sa << endl;
02228       }
02229     }
02230   } // end of r1 region
02231   exit(0);
02232
02233 }
```

Here is the call graph for this function:



4.26.3.18 void registerCarbonEvents ( )

call registerHarvesting(), registerDeathBiomass(), registerProducts() and registerTransports()

This function call registerHarvesting() (accounts for emissions from for. operations), registerDeathBiomass() (registers new stocks of death biomass), registerProducts() (registers new stock of products) and registerTransports() (accounts for emissions from transportation).

It pass to registerProducts():

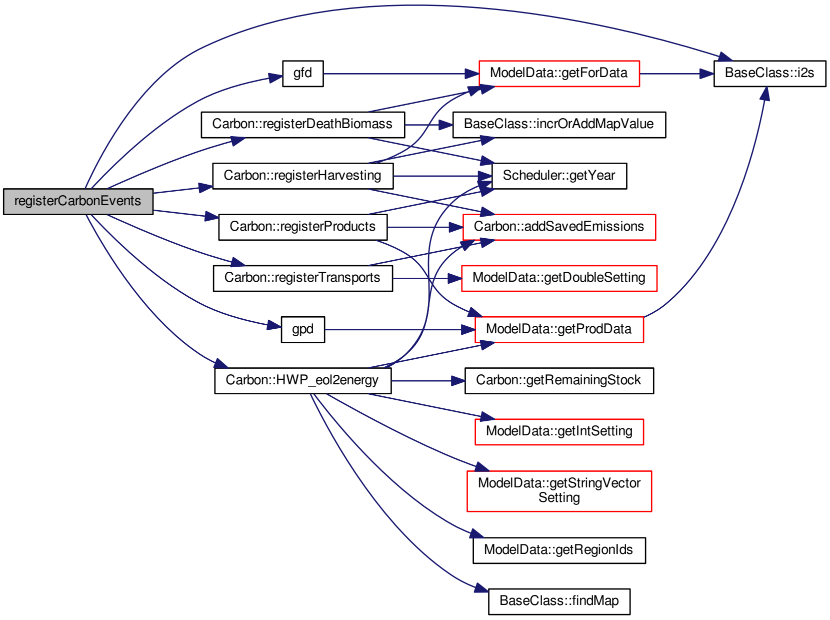
- for primary products, the primary products exported out of the country, but not those exported to other regions or used in the region as these are assumed to be totally transformed to secondary products;
- for secondary products, those produced in the region from locally or regionally imported primary product plus those secondary products imported from other regions, less those exported to other regions. It doesn't include the secondary products imported from abroad the country.

Definition at line 1959 of file ModelCoreSpatial.cpp.

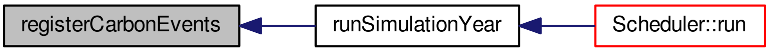
Referenced by runSimulationYear().

```
01959                                       {
01960
01961   //void                  registerHarvesting(const int & regId, const string & fType, const double &
     value); ///< register the harvesting of trees -> cumEmittedForOper
01962   //void                  registerDeathBiomass(const double &value, const int & regId, const string
     &fType);
01963   //void                  registerProducts(const double &value, const int & regId, const string
     &productName);
01964   //void                  registerTransports(const double &distQ, const int & regId);
01965
01966   for(uint i=0;i<regIds2.size();i++){
01967     for(uint j=0;j<fTypes.size();j++){
01968       double deathBiomass = gfd("vMort",regIds2[i],fTypes[j],
DIAM_ALL,DATA_NOW);
      double harvesting = gfd("hV",regIds2[i],fTypes[j],DIAM_ALL,
DATA_NOW);
      MTHREAD->CBAL->registerDeathBiomass(deathBiomass,
regIds2[i], fTypes[j]);  // register new stock
      MTHREAD->CBAL->registerHarvesting(harvesting,
regIds2[i], fTypes[j]);       // account for emissions. Added 201500715: it also moves the
 extra biomass to the death biomass pool
    }

    for(uint p=0;p<priProducts.size();p++){
      // for the primary products we consider only the exports as the domestic consumption is entirelly
 transformed in secondary products
      double int_exports = gpd("sa",regIds2[i],priProducts[p],
DATA_NOW);
      MTHREAD->CBAL->registerProducts(int_exports,
regIds2[i], priProducts[p]);  // register new stock
    }
    for(uint p=0;p<secProducts.size();p++){
      // for the tranformed product we skip those that are imported, hence derived from other forest
 systems
      // but we consider those coming from other regions
      double consumption = gpd("dl",regIds2[i],secProducts[p],
DATA_NOW); // dl = sl + net regional imports
      MTHREAD->CBAL->registerProducts(consumption,
regIds2[i], secProducts[p]); // register new stock
    }

  }
  for (uint r1=0;r1<l2r.size();r1++){
    for (uint r2=0;r2<l2r[r1].size();r2++){
      int rfrom= l2r[r1][r2];
      double distQProd = 0.0;
      for (uint r3=0;r3<l2r[r1].size();r3++){
        int rto = l2r[r1][r3];
        double dist = gpd("dist",rfrom,"",DATA_NOW,i2s(rto)); //km
        for(uint p=0;p<allProducts.size();p++){
          distQProd += dist*gpd("rt",rfrom,allProducts[p],DATA_NOW,
i2s(rto)); //km*Mm^3
        }
      }
      MTHREAD->CBAL->registerTransports(distQProd, rfrom);
    }
  }
  MTHREAD->CBAL->HWP_eol2energy(); // used to compute the energy substitution from
 hwp that reach the end of life and doesn't go to landfil. Previously the energy substitution was computed
 in registerProducts(), that is at the time when the product was produced.

}
```

Here is the call graph for this function:


registerCarbonEvents
gfd
Carbon::registerDeathBiomass
Carbon::registerHarvesting
Carbon::registerProducts
Carbon::registerTransports
gpd
Carbon::HWP_eol2energy
ModelData::getForData
BaseClass::incrOrAddMapValue
Scheduler::getYear
Carbon::addSavedEmissions
ModelData::getDoubleSetting
ModelData::getProdData
Carbon::getRemainingStock
ModelData::getIntSetting
ModelData::getStringVector Setting
ModelData::getRegionIds
BaseClass::findMap
BaseClass::i2s

Here is the caller graph for this function:



4.26.3.19 void resetPixelValues ( )

swap volumes->lagged_volumes and reset the other pixel vectors

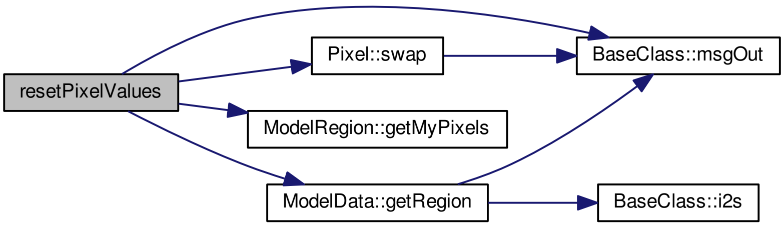
Definition at line 1509 of file ModelCoreSpatial.cpp.

Referenced by runInitPeriod(), and runSimulationYear().

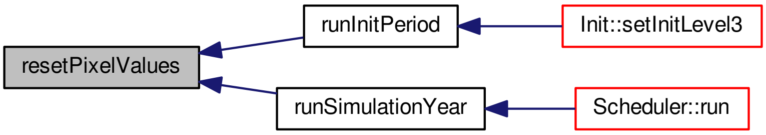
```
01509                                  {
01510   msgOut(MSG_INFO, "Starting resetting pixel level values");
01511   for(uint r2= 0; r2<regIds2.size();r2++){
01512     int regId = regIds2[r2];
01513     regPx = MTHREAD->MD->getRegion(regId)->getMyPixels();
01514     for (uint p=0;p<regPx.size();p++){
01515       Pixel* px = regPx[p];
01516       px->swap(VAR_VOL);  // vol_l = vol
01517       px->swap(VAR_AREA); // area_l = area
01518       // 20121108 BUG! Solved, used empty (just return true if the vector is empty) instead of clear (it
       actually clears the vector)
01519       px->vol.clear(); // by ft,dc
01520       px->area = px->area_l; // ATTENTION, DIFFERENT FROM THE OTHERS. Here it is not cleared, it
       is assigned the previous year as default
01521       /*px->area.clear(); // by ft,dc*/
```

```
      px->hArea.clear(); // by ft, dc
      //px->regArea.clear(); // by year, ft NO, this one is a map, it doesn't need to be changed
      px->hVol.clear(); // by ft, dc
      px->hVol_byPrd.clear(); // by ft, dc, pp
      //px->in.clear(); // by pp
      //px->hr.clear(); // by pp
      px->vReg.clear(); // by ft
      px->expectedReturns.clear(); // by ft

      px->beta.clear();
      px->mort.clear();
      px->tp.clear();
      px->cumTp.clear();
      px->vHa.clear();
      px->cumTp_exp.clear();
      px->vHa_exp.clear();
      px->cumAlive.clear();
      px->cumAlive_exp.clear();
      px->vMort.clear();
      //std::fill(rpx[j]->vMort.begin(), rpx[j]->vMort.end(), 0.0);

    }
  }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.26.3.20 void runBiologicalModule ( )

computes hV, hArea and new vol at end of year

ModelCoreSpatial::runBiologicalModule.

Changes in Area: dc area_l area diff 0 -------> +regArea -areaFirstProdClass (areaMovingUp_00) 15 -------> +areaFirstPrClass -hArea_15 -areaMovingUp_15 25 -------> +areaMovingUp15 - hArea_25 - areaMovingUp_25

35 -------—> +areaMovingUp25 - hArea_35 - areaMovingUp_35 ... 95 -------—> +areaMovingUp85 - hArea_95 - areaMovingUp_95 105 -------—> +areaMovingUp95 - hArea_105

note: regArea is computed in the management module, not here. Further, regArea is already the net one of forest area changes

Definition at line 475 of file ModelCoreSpatial.cpp.

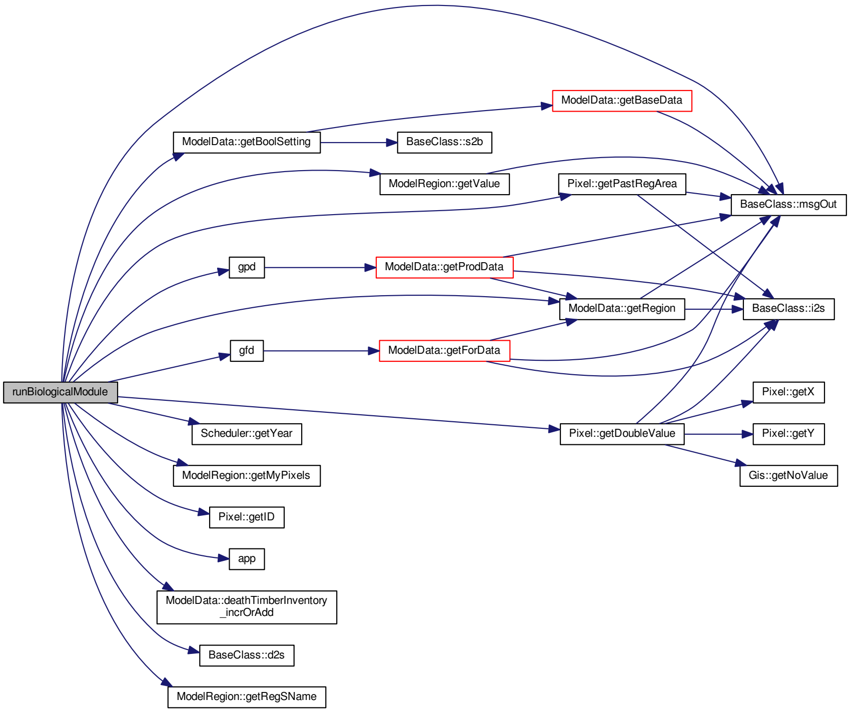
Referenced by runInitPeriod(), and runSimulationYear().

```
00475                                      {
00476
00477   msgOut(MSG_INFO, "Starting resource module..");
00478   int thisYear = MTHREAD->SCD->getYear();
00479   bool useDeathTimber = MD->getBoolSetting("useDeathTimber");
00480
00481   for(uint i=0;i<regIds2.size();i++){
00482     int r2 = regIds2[i];
00483     int regId = r2;
00484     ModelRegion* REG = MTHREAD->MD->getRegion(r2);
00485     //Gis* GIS = MTHREAD->GIS;
00486     regPx = REG->getMyPixels();
00487     double shareMortalityUsableTimber;
00488     if(useDeathTimber){
00489         shareMortalityUsableTimber = gfd("shareMortalityUsableTimber",r2,"","");
00490     } else {
00491         shareMortalityUsableTimber = 0.0;
00492     }
00493
00494     for (uint p=0;p<regPx.size();p++){
00495       Pixel* px = regPx[p];
00496
00497       double pxId = px->getID();
00498       //if (pxId == 3550.0){
00499       //    cout << "got the pixel" << endl;
00500       //}
00501       //px->expectedReturns.clear();
00502       for(uint j=0;j<fTypes.size();j++){
00503         string ft = fTypes[j];
00504         double pxArea_debug     = px->getDoubleValue("forArea_"+ft, true);
00505         vector <double>            hV_byDiam;
00506         vector < vector <double> > hV_byDiamAndPrd;
00507         vector <double> hArea_byDc;
00508         vector <double> newVol_byDiam;
00509         vector <double> vMort_byDc;
00510         vector <double> areasMovingUp(dClasses.size(), 0.0);
00511         double areaFirstProdClass;
00512
00513
00514         // A - COMPUTING THE REGENERATION..
00515         // if we are in a year where the time of passage has not yet been reached
00516         // for the specific i,e,l then we use the exogenous Vregen, otherwise we
00517         // calculate it
00518         //if ( not scen("fxVreg") ,
00519         //  loop( (i,essence,lambda),
00520         //    if( ord(t)>=(tp_u1(i,essence,lambda)+2),
00521         //
       Vregen(i,lambda,essence,t)=regArea(i,essence,lambda,t-tp_u1(i,essence,lambda))*volHa_u1(i,essence,lambda)/1000000   ;
00522         //    );
00523         //  );
00524         //);
00525         int tp_u0 = px->tp.at(j).at(0); // time of passage to reach the first production diameter class
       // bug 20140318, added ceil. 20140318 removed it.. model did go crazy with it
00526         if(thisYear == secondYear){
00527             px->initialDc0Area.push_back(px->area_l.at(j).at(0));
00528         }
00529         if(regType != "fixed" && (thisYear-secondYear) >= tp_u0 ) { // T.O.D.O to be
       checked -> 20121109 OK
00530           double pastRegArea = px->getPastRegArea(j,thisYear-tp_u0);
00531           double availableArea = px->area_l.at(j).at(0);
00532           //double entryVolHa = gfd("entryVolHa",regId,ft,"");
00533           double vHa = px->vHa.at(j).at(1);
00534           //attention that at times could take the wrong pastRegArea if tp change too suddenly as in some
       "strange" scenarios
00535           if (oldVol2AreaMethod){
00536             areaFirstProdClass = pastRegArea;
00537           } else {
00538             areaFirstProdClass = min(availableArea, pastRegArea); // this is just a start and will need to
       include the last year area
00539           }
00540           px->vReg.push_back(areaFirstProdClass*vHa/1000000.0); // TO.DO: check the 1000000. Should be
       ok, as area in ha vol in Mm^3
```

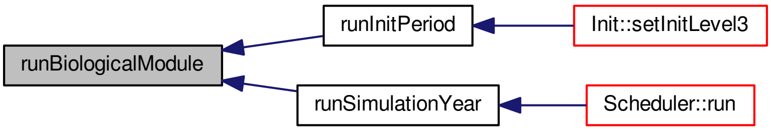
```
  //if (pxId == 3550.0 && j==3){
  //    cout << "got the pixel" << endl;
  //}
  #ifdef QT_DEBUG
  if (areaFirstProdClass < 0.0){
     //msgOut(MSG_CRITICAL_ERROR,"Negative regeneration volumes in endogenous regeneration");
  }
  if ( (availableArea-pastRegArea) < -0.00000001     ){
     // in a very rare cases tp change first in a direction and then in the other, so that the wrong past regeneration area
     // is picken up.
     //msgOut(MSG_CRITICAL_ERROR,"Upgrading from dc0 more area than the available one in endogenous regeneration");
  }
  #endif
} else {
  double regionArea = REG->getValue("forArea_"+ft,OP_SUM);
  double pxArea     = px->getDoubleValue("forArea_"+ft, true); // 20121109 bug solved (add get zero for not data)
  double regRegVolumes = gfd("vReg",r2,ft,"");
  double newVReg = regionArea ? regRegVolumes*pxArea/regionArea : 0.0;
  px->vReg.push_back(newVReg); // 20121108 BUG !!! solved // as now we have the area we could also use here entryVolHa
  // only a share of the exogenous area goes up, the regeneration one doesn't yet reach tp0:
  // areaFirstProdClass = (1.0 / px->tp.at(j).at(0) ) * px->area_l.at(j).at(0);
  areaFirstProdClass = (1.0 / ((double) tp_u0) ) * px->initialDc0Area.at(j);
  // in the exogenous period we are exogenously upgrading u0->u1 some areas but, as we do not have the regeneration
  // are corresponding to that we have also to manually add it to u0
  //px->area_l.at(j).at(0) += areaFirstProdClass;
  //areaFirstProdClass = entryVolHa ? newVReg*1000000 /entryVolHa:0.0;
  //if (pxId == 3550.0 && j==3){
  //    cout << "got the pixel" << endl;
  //}

  #ifdef QT_DEBUG
  if (areaFirstProdClass<0.0){
    // msgOut(MSG_CRITICAL_ERROR,"Negative regeneration volumes in exogenous regeneration");
  }
  if (areaFirstProdClass > px->area_l.at(j).at(0)){
     //msgOut(MSG_CRITICAL_ERROR,"Moving up area higher than available area in exogenous regeneration !");
  }
  #endif
  // vReg and entryVolHa are NOT the same thing. vReg is the yearly regeneration volumes
  // for the whole region. We can use them when we don't know the harvested area
  // entryVolHa can lead to vReg calculation only when we know the regeneration area. So in the
  // first years we use vReg and subsequently the endogenous one.
}

//double harvestedArea = 0;



for (uint u=0; u<dClasses.size(); u++){
  string dc = dClasses[u];
  double hr =0;
  //double pastYearVol_reg = u ? gfd("vol",r2,ft,dc,thisYear-1): 0;
  double pastYearVol = px->vol_l.at(j).at(u);
  vector <double> hV_byPrd;
  vector <double> hr_byPrd;

  // harvesting rate & volumes...
  // hr is by region.. no reasons in one pixel the RATE of harvesting will be different than in an other pixel
  //hr(u,i,essence,lambda,t) =    sum(p_pr, prov(u,essence,lambda,p_pr)*RPAR('st',i,p_pr,t)/In(i,p_pr,t));
  //hV(u,i,essence,lambda,t) = hr(u,i,essence,lambda,t) * V(u,i,lambda,essence,t-1);
  //hV_byPrd(u,i,essence,lambda,p_pr,t) = prov(u,essence,lambda,p_pr)*(RPAR('st',i,p_pr,t)/In(i,p_pr,t))*V(u,i,lambda,essence,t-1);
  for(uint pp=0;pp<priProducts.size();pp++){
    double st = gpd("stFromHarvesting",r2,priProducts[pp]);
    double in = gpd("in",r2,priProducts[pp]);
    double hr_pr = in ? app(priProducts[pp],ft,dc)*st/in : 0.0;
    hr_byPrd.push_back( hr_pr);
    hr += hr_pr;
  }

  // adjusting for overharvesting..
  // 20160204: inserted to account that we let supply to be marginally higher than in in the mamarket module, to let the solver solving
  double origHr = hr;
  hr = min(1.0,hr);
  for(uint pp=0;pp<priProducts.size();pp++){
    double hr_pr = origHr ? hr_byPrd[pp] * min(1.0,1.0/origHr) : 0.0;
    hV_byPrd.push_back( hr_pr*pastYearVol*px->avalCoef);
  }
```

```

          double hV = hr*pastYearVol*px->avalCoef;


          hV_byDiam.push_back(hV);
          hV_byDiamAndPrd.push_back(hV_byPrd);

          // post harvesting remained volumes computation..
          // loop(u$(ord(u)=1),
          //  first diameter class, no harvesting and fixed regenaration..
          //  V(u,i,lambda,essence,t)=(1-1/(tp(u,i,lambda,essence))-mort(u,i,lambda,essence)
 )*V(u,i,lambda,essence,t-1)
          //                           +Vregen(i,lambda,essence,t);
          // );
          // loop(u$(ord(u)>1),
          //  generic case..
          //  V(u,i,lambda,essence,t)=((1-1/(tp(u,i,lambda,essence))
          //                           -mort(u,i,lambda,essence) -
 hr(u,i,essence,lambda,t))*V(u,i,lambda,essence,t-1)
          //
 +(1/(tp(u-1,i,lambda,essence)))*beta(u,i,lambda,essence)*V(u-1,i,lambda,essence,t-1));
          double vol;
          double tp          = px->tp.at(j).at(u); //gfd("tp",regId,ft,dc);
          double mort        = px->mort.at(j).at(u); //gfd("mortCoef",regId,ft,dc);
          double vReg        = px->vReg.at(j); //gfd("vReg",regId,ft,""); // Taking it from the memory
 database as we could be in a fixed vReg scenario and not having calculated it from above!
          double beta        = px->beta.at(j).at(u); //gfd("betaCoef",regId,ft,dc);
          //double hv2fa       = gfd("hv2fa",regId,ft,dc);
          double vHa         = px->vHa.at(j).at(u); //gfd("vHa",regId,ft,dc);
          double finalHarvestFlag = gfd("finalHarvestFlag",regId,ft,dc);

          double vMort       = mort*pastYearVol;

          vMort_byDc.push_back(vMort);

          if(useDeathTimber){
            iisskey key(thisYear,r2,ft,dc);
            MD->deathTimberInventory_incrOrAdd(key,vMort*
shareMortalityUsableTimber);
          }

          if(u==0){
            vol = 0.0;
          }else if(u==1){
            vol = max(0.0,(1-1/tp-mort))*pastYearVol+vReg; //Antonello, "bug" fixed 20160203: In case of
 very strong mortality this quantity (that doesn't include harvesting) could be negative!
            double debug = vol;
            #ifdef QT_DEBUG
            if ((1-1/tp-mort)<0.0){
              msgOut(MSG_DEBUG,"The sum of leaving trres and mortality would have lead to
 nevative volume if we didn't put a max. 1/tp: "+d2s(1/tp)+",  mort: "+d2s(mort)+", total coeff: "+
d2s((1-1/tp-mort))+" ");
            }
            #endif
          } else {
            // time of passage and volume of smaller diameter class
            double inc = (u==dClasses.size()-1)?0:1./tp; // we exclude the possibility for trees in
 the last diameter class to move to an upper class
            double tp_1 = px->tp.at(j).at(u-1); //gfd("tp",regId,ft,dClasses[u-1]);
            double pastYearVol_1 = px->vol_l.at(j).at(u-1); //
gfd("vol",regId,ft,dClasses[u-1],thisYear-1);
            //vol = max(0.0,(1-inc-mort-hr)*pastYearVol+(1/tp_1)*beta*pastYearVol_1);
            vol = max(0.0,(1-inc-mort)*pastYearVol-hV+(1/tp_1)*beta*pastYearVol_1); // I can't use any more
 hr as it is the harvesting rate over the available volumes, not the whole ones
            #ifdef QT_DEBUG
            if ((1-inc-mort)*pastYearVol-hV+(1/tp_1)*beta*pastYearVol_1 < 0){
              double realVolumes = (1-inc-mort)*pastYearVol-hV+(1/tp_1)*beta*pastYearVol_1;
              msgOut(MSG_DEBUG,"Negative real volumes ("+d2s(realVolumes)+"), possibly
 because of little bit larger bounds in the market module to avoid zeros. Volumes in the resource module set
 back to zero, so it should be ok.");
            }
            #endif
          }
          if(u != 0){ // this if is required to avoid a 0/0 and na error that then propage also in vSum()
            double inc = (u==dClasses.size()-1)?0:1.0/tp; // we exclude the possibility for trees
 in the last diameter class to move to an upper class
            double volumesMovingUp = inc*pastYearVol;
            double pastArea = px->area_l.at(j).at(u);

            areasMovingUp.at(u) = inc*pastArea;

            if(oldVol2AreaMethod) {
              hArea_byDc.push_back(finalHarvestFlag*1000000*hV/vHa); // volumes are in Mm^3, area in ha,
 vHa in m^3/ha
            } else {
              double finalHarvestedVolumes = finalHarvestFlag* hV;
              double finalHarvestedRate = pastYearVol?finalHarvestedVolumes/pastYearVol:0.0; // Here we
want the harvested rate over the whole volumes, not just the available ones, so we don't need to multiply to px->avalCoef
       #ifdef QT_DEBUG
       if (finalHarvestedRate > 1.0){
        msgOut(MSG_CRITICAL_ERROR,"Negative final harvested rate.");
       }
       #endif
       hArea_byDc.push_back(finalHarvestedRate*pastArea); // volumes are in Mm^3, area in ha, vHa in m^3/ha
      }
      px->area.at(j).at(u) = max(0.0, px->area_l.at(j).at(u) - areasMovingUp.at(u) + areasMovingUp.at(u-1) - hArea_byDc.at(u));
      #ifdef QT_DEBUG
      if ((px->area_l.at(j).at(u) - areasMovingUp.at(u) + areasMovingUp.at(u-1) - hArea_byDc.at(u))< 0.0){
       msgOut(MSG_DEBUG,"If not for a max, we would have had a negative area ("+ d2s(px->area_l.at(j).at(u) - areasMovingUp.at(u) + areasMovingUp.at(u-1) - hArea_byDc.at(u))+" ha).");
      }
      #endif
     } else {
      areasMovingUp.at(u) = areaFirstProdClass;
      hArea_byDc.push_back(0.);
      px->area.at(j).at(u) = px->area_l.at(j).at(u) - areasMovingUp.at(u) - hArea_byDc.at(u);
      //if (pxId == 3550.0 && j==3){
      //    cout << "got the pixel" << endl;
      //}
     }
     newVol_byDiam.push_back(vol);
     #ifdef QT_DEBUG
     if(px->area.at(j).at(u)< 0.0 || areasMovingUp.at(u) < 0.0 || hArea_byDc.at(u) < 0.0 ){
      msgOut(MSG_CRITICAL_ERROR, "Negative values in runBiologicalModel");
     }
     #endif

     //double debug = hv2fa*hr*pastYearVol*100;
     //cout << "regId|ft|dc| debug | freeArea: " << r2 << "|"<<ft<<"|"<<dc<<"| "<< debug << " | " << freeArea_byU << endl;

     //sfd(hr,"hr",regId,ft,dc);
     //sfd(hV,"hV",regId,ft,dc);
     //sfd(vol,"vol",regId,ft,dc);

     //sfd(freeArea_byU,"harvestedArea",regId,ft,dc,DATA_NOW,true);
    } // end foreach diameter classes
    px->hVol.push_back(hV_byDiam);
    px->hVol_byPrd.push_back(hV_byDiamAndPrd);
    px->hArea.push_back(hArea_byDc);
    px->vol.push_back(newVol_byDiam);
    px->vMort.push_back(vMort_byDc);


    #ifdef QT_DEBUG
    for (uint u=1; u<dClasses.size(); u++){
     double volMort = vMort_byDc[u];
     double harvVol = hV_byDiam[u];
     double vol_new = newVol_byDiam[u];
     double vol_lagged = px->vol_l.at(j).at(u);
     double gain = vol_new - (vol_lagged-harvVol-volMort);
     if (volMort > vol_lagged){
      msgOut(MSG_CRITICAL_ERROR,"mort vol > lagged volumes ?");
     }
    }
    #endif
   } // end of each forest type
  } // end of each pixel

  #ifdef QT_DEBUG
  // checking that in a region the total hVol is equal to the st for each products. 20150122 Test passed with the new availCoef
  double sumSt = 0.0;
  double sumHv = 0.0;
  for(uint pp=0;pp<priProducts.size();pp++){
   sumSt += gpd("stFromHarvesting",r2,priProducts[pp]);
  }
  for (uint p=0;p<regPx.size();p++){
   for(uint j=0;j<fTypes.size();j++){
    for (uint u=0; u<dClasses.size(); u++){
     for(uint pp=0;pp<priProducts.size();pp++){
      // by ft, dc, pp
      sumHv += regPx[p]->hVol_byPrd[j][u][pp];
     }
    }
   }
  }
  if(abs(sumSt-sumHv) > 0.000001){
        msgOut(MSG_DEBUG, "St and harvested volumes diverge in region "+REG->
   getRegSName()+". St: "+d2s(sumSt)+" hV: "+d2s(sumHv));
      }
      #endif
    } // end of each region

}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.26.3.21 void runInitPeriod ( )

< cashe things like first year, second year, dClasses...

< compute px volumes vol for 2005 (including exogenous loaded volumes)

< inside it uses first year, second year

< 2005->2006

< swap volumes->lagged_volumes and reset the other pixel vectors

< compute pixel tp, meta and mort

< in=f(vol_t-1)

< compute cumTp_exp, vHa_exp, vHa

< compute px->area for each ft and dc (including exogenous loaded areas)

< update the forArea_{ft} layer on each pixel as old value-hArea+regArea

< update (if the layer exists) other gis-based data, as volumes and expected returns, taking them from the data in the px object

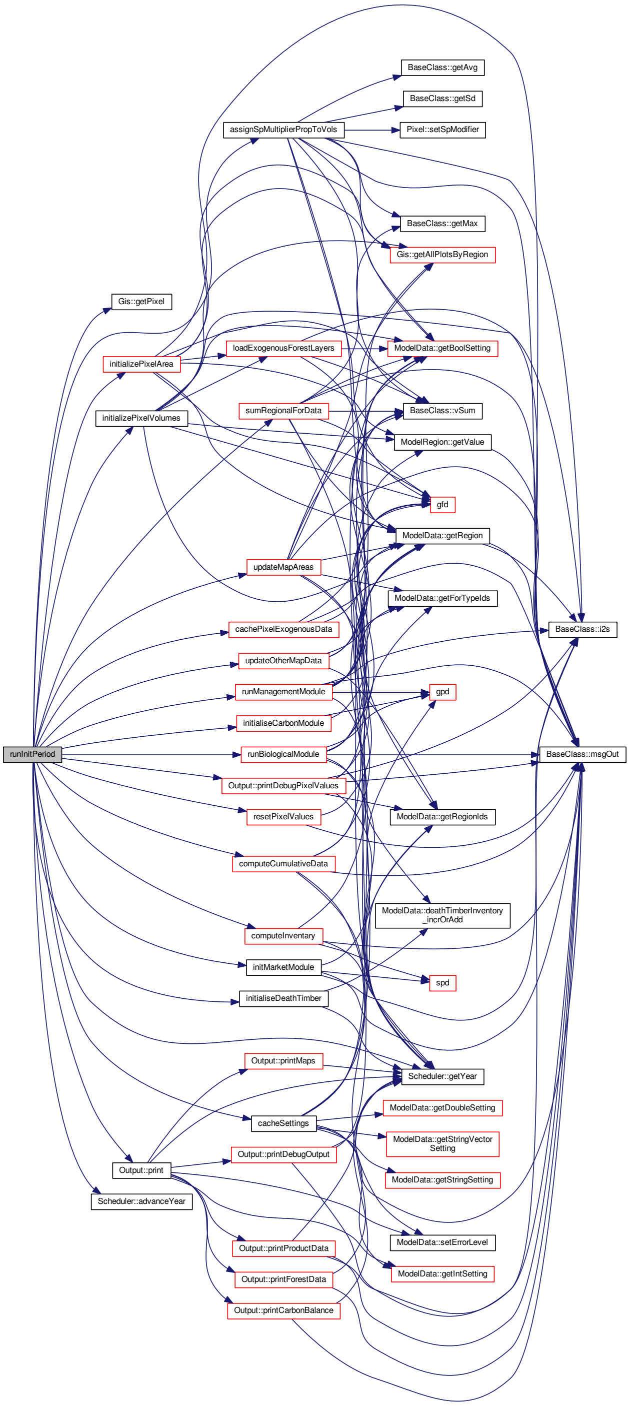
< only for printing stats as forest data is never used at regional level

Definition at line 46 of file ModelCoreSpatial.cpp.

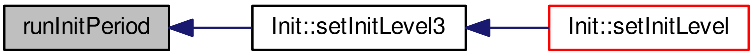
Referenced by Init::setInitLevel3().

```
00046                               {
00047   Pixel* debug = MTHREAD->GIS->getPixel(20798);
00048   cacheSettings();                ///< cashe things like first year, second year, dClasses...
00049   initializePixelVolumes();       ///< compute px volumes vol for 2005 (including
       exogenous loaded volumes)
00050   assignSpMultiplierPropToVols(); // assign the spatial multiplier (used in the
       time of return) based no more on a Normal distribution but on the volumes present in the pixel: more
       volume, more the pixel is fit for the ft
00051   initMarketModule();             ///< inside it uses first year, second year
00052   initialiseDeathTimber();
00053   MTHREAD->DO->print();
00054   MTHREAD->SCD->advanceYear();    ///< 2005->2006
00055   int thisYear = MTHREAD->SCD->getYear(); // for debugging
00056   resetPixelValues();             ///< swap volumes->lagged_volumes and reset the other
       pixel vectors
00057   cachePixelExogenousData();      ///< compute pixel tp, meta and mort
00058   computeInventary();             ///< in=f(vol_t-1)
00059 //printDebugInitRegionalValues();
00060   computeCumulativeData();        ///< compute cumTp_exp, vHa_exp, vHa
00061   initializePixelArea();          ///< compute px->area for each ft and dc (including
       exogenous loaded areas)
00062   runBiologicalModule();
00063   runManagementModule();
00064   MTHREAD->DO->printDebugPixelValues(); // uncomment to enable pixel-level
       debugging
00065   updateMapAreas();               ///< update the forArea_{ft} layer on each pixel as old
       value-hArea+regArea
00066   updateOtherMapData();           ///< update (if the layer exists) other gis-based data,
       as volumes and expected returns, taking them from the data in the px object
00067   sumRegionalForData();           ///< only for printing stats as forest data is never
       used at regional level
00068   initialiseCarbonModule();
00069
00070
00071   MTHREAD->DO->print();
00072 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.26.3.22 void runManagementModule ( )

computes regArea and expectedReturns

Definition at line 777 of file ModelCoreSpatial.cpp.

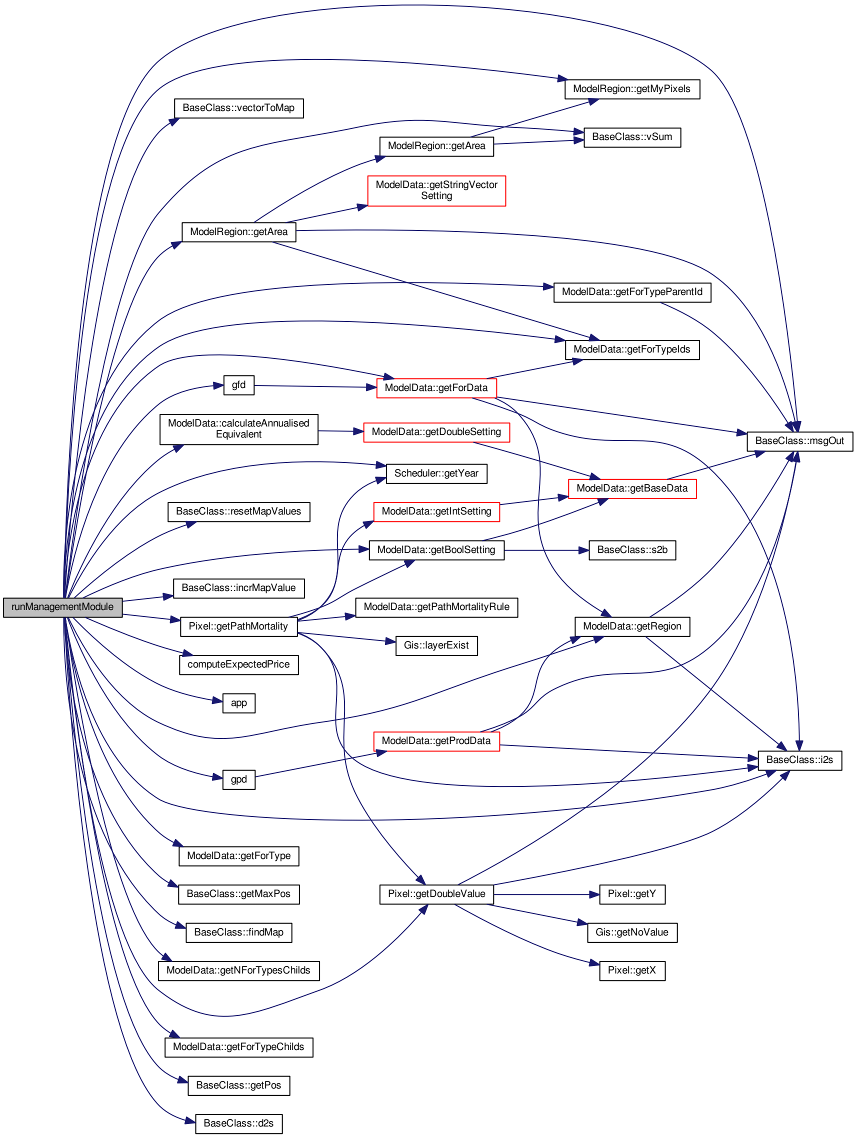
Referenced by runInitPeriod(), and runSimulationYear().

```
00777                                     {
00778  msgOut(MSG_INFO, "Starting management module..");
00779  vector<string> allFTypes = MTHREAD->MD->getForTypeIds(true);
00780  map<string,double> hAreaByFTypeGroup = vectorToMap(allFTypes,0.0);
00781  int thisYear = MTHREAD->SCD->getYear();
00782
00783  // Post optimisation management mobule..
00784  for(uint i=0;i<regIds2.size();i++){
00785    int r2 = regIds2[i];
00786    int regId = r2;
00787    ModelRegion* REG = MTHREAD->MD->getRegion(r2);
00788    regPx = REG->getMyPixels();
00789
00790    // Dealing with area change..
00791    double fArea_reg     = REG->getArea();
00792    double fArea_diff    = 0.0;
00793    double fArea_reldiff = 0.0;
00794    if(forestAreaChangeMethod=="relative"){
00795      fArea_reldiff = gfd("forestChangeAreaIncrementsRel",r2,"","",DATA_NOW);
00796      fArea_diff    = fArea_reg * fArea_reldiff;
00797    } else if (forestAreaChangeMethod=="absolute"){
00798      fArea_diff    = gfd("forestChangeAreaIncrementsHa",r2,"","",DATA_NOW);
00799      //fArea_reldiff = fArea_diff / fArea_reg;
00800    }
00801    double regHArea = 0.0; // for the warning
00802
00803
00804
00805
00806    for (uint p=0;p<regPx.size();p++){
00807      Pixel* px = regPx[p];
00808      px->expectedReturns.clear();
00809      px->expectedReturnsNotCorrByRa.clear(); // BUG discovered 20160825
00810      resetMapValues(hAreaByFTypeGroup,0.0);
00811      double totalHarvestedArea = vSum(px->hArea); // still need to remove the forest decrease
      areas..
00812      vector<double> thisYearRegAreas(fTypes.size(),0.0); // initialize a vector of fTypes.size()
      zeros.
00813      vector<double> expectedReturns(fTypes.size(),0.0);  // uncorrected expected returns (without
      considering transaction costs). These are in form of eai
00814
00815      double fArea_px = vSum(px->area);
00816      double fArea_diff_px = fArea_px * fArea_diff/ fArea_reg;
00817      double fArea_incr = max(0.0,fArea_diff_px);
00818      double fArea_decr = - min(0.0,fArea_diff_px);
00819      double fArea_decr_rel = totalHarvestedArea?min(1.0,fArea_decr/totalHarvestedArea):0.0;
00820      regHArea += totalHarvestedArea;
00821      totalHarvestedArea = totalHarvestedArea *(1-fArea_decr_rel);
00822
00823
00824      // A - Computing the harvestingArea by parent ft group (for the allocation according to the prob of
      presence):
00825      for(uint j=0;j<fTypes.size();j++){
00826        string ft = fTypes[j];
00827        string parentFt = MTHREAD->MD->getForTypeParentId(ft);
```

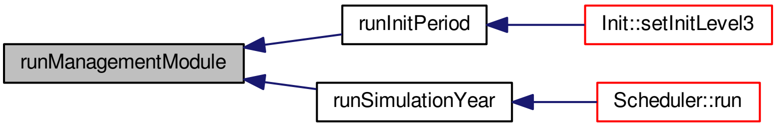
```
       double hAreaThisFt=vSum(px->hArea.at(j))*(1-fArea_decr_rel);
       incrMapValue(hAreaByFTypeGroup,parentFt,hAreaThisFt); // increment the parent ft of the
harvested area, neeed for assigning the frequences (prob. of presence)
     }

     // B - Computing the uncorrected expected returns (without considering transaction costs)
     // 20120910, Antonello: changed.. calculating the expected returns also for fixed and fromHrLevel
regeneration (then not used but gives indication)
     // calculating the expected returns..
     //  loop ( (u,i,essence,lambda,p_pr),
     //    if (sum(u2, hV(u2,i,essence,lambda,t))= 0,
     //      expRetPondCoef(u,i,essence,lambda,p_pr) = 0;
     //    else
     //      expRetPondCoef(u,i,essence,lambda,p_pr) = hV_byPrd(u,i,essence,lambda,p_pr,t)/ sum(u2,
hV(u2,i,essence,lambda,t));
     //    );
     //  );
     //  expReturns(i,essence,lambda) = sum( (u,p_pr),
     //            RPAR("pl",i,p_pr,t)*hv2fa(i,essence,lambda,u)*(1/df_byFT(u,i,lambda,essence))*
// df_byFT(u,i,lambda,essence)
     //            expRetPondCoef(u,i,essence,lambda,p_pr)
     //            );
     for(uint j=0;j<fTypes.size();j++){
       string ft = fTypes[j];
       double expReturns = 0.;
       int optDc = 0; // "optimal diameter class", the one on which the expected returns are computed
       for (uint u=0; u<dClasses.size(); u++){
         string dc = dClasses[u];
         double vHa          = px->vHa_exp.at(j).at(u);
         double finalHarvestFlag = gfd("finalHarvestFlag",regId,ft,dc);
         double cumTp_u = px->cumTp_exp.at(j).at(u);
         for (uint pp=0;pp<priProducts.size();pp++){
           double pl             = gpd("pl",regId,priProducts[pp]); // note that this is the
OBSERVED price. If we call it at current year+cumTp_u we would have the expected price. But we would first
have to compute it, as pw is weigthed price world-local and we don't have local price for the future. DONE
20141202 ;-)
           double worldCurPrice = gpd("pl",WL2,priProducts[pp]);
           double worldFutPrice = gpd("pl",WL2,priProducts[pp],thisYear+cumTp_u);
           double sl             = gpd("sl",regId,priProducts[pp]);
           double sa             = gpd("sa",regId,priProducts[pp]);
           double pw_exp         = computeExpectedPrice(pl, worldCurPrice,
worldFutPrice, sl, sa, px->expTypePrices); //20141030: added the expected price!
           double raw_amount = finalHarvestFlag*pw_exp*vHa*app(priProducts[pp],ft,dc); //
B.U.G. 20121126, it was missing app(pp,ft,dc) !!
           double anualised_amount =  MD->calculateAnnualisedEquivalent(
raw_amount,cumTp_u);
           if (anualised_amount>expReturns) {
             expReturns=anualised_amount;
             optDc = u;
           }
         }
       }
       px->expectedReturnsNotCorrByRa.push_back(expReturns);
       if(MD->getBoolSetting("heterogeneousRiskAversion")){
         double ra = px->getDoubleValue("ra");
         double cumMort = 1-px->cumAlive_exp.at(j).at(optDc);
         //cout << px->getID() << "\t" << ft << "\t\t" << "optDc" << optDc << "\t" << cumMort << endl;
         double origExpReturns = expReturns;
         expReturns = origExpReturns * (1.0 - ra*cumMort);
       }
       px->expectedReturns.push_back(expReturns);
       expectedReturns.at(j) = expReturns;
     } // end foreach forest type

     for(uint j=0;j<fTypes.size();j++){
       string ft = fTypes[j];
       forType* thisFt = MTHREAD->MD->getForType(ft);

       double harvestedAreaForThisFT = vSum(px->hArea.at(j))*(1-fArea_decr_rel); //
gfd("harvestedArea",regId,ft,DIAM_ALL);
       vector<double> corrExpectedReturns(fTypes.size(),0.0); //  corrected expected returns
(considering transaction costs). These are in form of  NPV

       // C - Computing the corrected expected returns including transaction costs
       for(uint j2=0;j2<fTypes.size();j2++){
         string ft2 = fTypes[j2];
         double invTransCost = gfd("invTransCost",regId,ft,ft2,DATA_NOW);
         corrExpectedReturns[j2] = (expectedReturns[j2]/ir)-invTransCost; // changed 20150718: npv =
eai/ir + tr. cost // HUGE BUG 20151202: transaction costs should be REDUCED, not added to the npv...
       }

       //int highestReturnFtIndex = getMaxPos(corrExpectedReturns);

       // D - Assigning the Managed area
       // calculating freeArea at the end of the year and choosing the new regeneration area..
       //freeArea(i,essence,lambda) = sum(u,
hv2fa(i,essence,lambda,u)*hr(u,i,essence,lambda,t)*V(u,i,lambda,essence,t-1)*100);
```

```
        //if(scen("endVreg") ,
        //  regArea(i,essence,lambda,t) = freeArea(i,essence, lambda);   // here we could introduce in/out area from other land usages
        //else
        //  loop (i,
        //    loop( (essence,lambda),
        //      if ( expReturns(i,essence,lambda) = smax( (essence2,lambda2),expReturns(i,essence2,lambda2)),
        //        regArea (i,essence,lambda,t) =  sum( (essence2, lambda2), freeArea(i,essence2, lambda2) ) * mr;
        //      );
        //    );
        //    regArea(i,essence,lambda,t) = freeArea(i,essence, lambda)*(1-mr);   // here we could introduce in/out area from other land usages
        //  );
        //if (j==highestReturnFtIndex){
        //  thisYearRegAreas[j] += totalHarvestedArea*mr;
        //}
        // If I Implement this I'll have a minimal diff in total area.. why ?????

        double mr = MD->getForData("mr",regId,"","");
        thisYearRegAreas[getMaxPos(corrExpectedReturns)] += harvestedAreaForThisFT*mr;
        thisYearRegAreas[getMaxPos(expectedReturns)] += fArea_incr*mr/((double)fTypes.size()); // mr quota of new forest area assigned to highest expected returns ft (not considering transaction costs). Done for each forest types


        // E - Assigning unmanaged area
        //for(uint j2=0;j2<fTypes.size();j2++){
          if(natRegAllocation=="pp"){ // according to prob presence
            //string ft2 = fTypes[j2];
            string parentFt = MTHREAD->MD->getForTypeParentId(ft);
            double freq = rescaleFrequencies ? gfd("freq_norm",regId,parentFt,""):gfd("freq",regId,parentFt,""); // "probability of presence" for unmanaged forest, added 20140318
            double hAreaThisFtGroup = findMap(hAreaByFTypeGroup,parentFt);
            double hRatio = 1.0;
            if(hAreaThisFtGroup>0){
                //double harvestedAreaForThisFT2 = vSum(px->hArea.at(j2));
                hRatio = harvestedAreaForThisFT/hAreaThisFtGroup;
            } else {
                int nFtChilds = MTHREAD->MD->getNForTypesChilds(parentFt);
                hRatio = 1.0/nFtChilds;
            }
            thisYearRegAreas[j] +=  totalHarvestedArea*(1-mr)*freq*hRatio;
            thisYearRegAreas[j] +=  fArea_incr*(1-mr)*freq*hRatio; // non-managed quota of new forest area assigning proportionally on pp at sp group level
            //thisYearRegAreas[j2] +=  harvestedAreaForThisFT*(1-mr)*freq*hRatio;
          } else { // prob presence not used..

            // Accounting for mortality arising from pathogens. Assigning the area to siblings according to area..


            double mortRatePath = px->getPathMortality(ft, "0");
            if(mortRatePath > 0){

              string parentFt = MTHREAD->MD->getForTypeParentId(ft);
              vector <string> siblings = MTHREAD->MD->getForTypeChilds(parentFt);
              vector <double> siblingAreas;
              for(uint j2=0;j2<siblings.size();j2++){
                if(siblings[j2]==ft){
                  siblingAreas.push_back(0.0);
                } else {
                  string debug_sibling_ft = siblings[j2];
                  int debug_positin = getPos(debug_sibling_ft,fTypes);
                  double thisSiblingArea = vSum(px->area.at(getPos(siblings[j2],fTypes)));
                  siblingAreas.push_back(thisSiblingArea);
                }
              }
              double areaAllSiblings = vSum(siblingAreas);
              thisYearRegAreas[j] += harvestedAreaForThisFT*(1-mr)*(1-mortRatePath);

              if(areaAllSiblings>0.0){ // area of siblings is >0: we attribute the area from the pathogen induced mortality to the siblings proportionally to area..
                for(uint j2=0;j2<siblings.size();j2++){
//                  int debug1 =  getPos(siblings[j2],fTypes);
//                  double debug2= harvestedAreaForThisFT;
//                  double debug3 = 1.0-mr;
//                  double debug4 = mortRatePath;
//                  double debug5 = siblingAreas[j2];
//                  double debug6 = areaAllSiblings;
//                  double debug7 = harvestedAreaForThisFT*(1.0-mr)*(mortRatePath)*(siblingAreas[j2]/areaAllSiblings);
                  thisYearRegAreas[getPos(siblings[j2],fTypes)] += harvestedAreaForThisFT*(1.0-mr)*(mortRatePath)*(siblingAreas[j2]/areaAllSiblings);
                }
              } else if (siblings.size()>1) { // area of all siblings is 0, we just give them the mortality area in equal parts..
                for(uint j2=0;j2<siblings.size();j2++){
                  if (siblings[j2] != ft){
                    thisYearRegAreas[getPos(siblings[j2],fTypes)] += harvestedAreaForThisFT*(1.0-mr)*(mortRatePath)* 1.0 / (( (float) siblings.size())-1.0);
                  }
                }
              }
            } else { // mortRatePath == 0
              thisYearRegAreas[j] += harvestedAreaForThisFT*(1.0-mr);
            }

            // Allocating non-managed quota of new forest area to ft proportionally to the current area share by ft
            double newAreaThisFt = vSum(px->area) ? fArea_incr*(1-mr)*vSum(px->area.at(j))/vSum(px->area): 0.0;
            thisYearRegAreas[j] +=  newAreaThisFt;
            if(! (thisYearRegAreas[j] >= 0.0) ){
              msgOut(MSG_ERROR,"thisYearRegAreas[j] is not non negative (j: "+i2s(j)+", thisYearRegAreas[j]: "+i2s( thisYearRegAreas[j])+").");
            }
            //thisYearRegAreas[j2] += harvestedAreaForThisFT*(1-mr);
          }
        //}
      } // end for each forest type

      // adding regeneration area to the fist (00) diameter class
      for(uint j=0;j<fTypes.size();j++){
        px->area.at(j).at(0) += thisYearRegAreas.at(j);
      }

      #ifdef QT_DEBUG
      double totalRegArea = vSum(thisYearRegAreas);
      if (! (totalRegArea==0.0 && totalHarvestedArea==0.0)){
        double ratio = totalRegArea / totalHarvestedArea ;
        if(rescaleFrequencies && (ratio < 0.99999999999 || ratio > 1.00000000001) ) {
          msgOut(MSG_CRITICAL_ERROR, "Sum of regeneration areas not equal to sum of harvested area in runManagementModel()!");
        }
      }
      #endif
      px->regArea.insert(pair <int, vector<double> > (MTHREAD->SCD->getYear(), thisYearRegAreas));
    } // end of each pixel
    if (-fArea_diff > regHArea){
      msgOut(MSG_WARNING,"In region "+ i2s(regId) + " the exogenous area decrement ("+d2s(-fArea_diff) +" ha) is higger than the harvesting ("+ d2s(regHArea) +" ha). Ratio forced to 1.");
    }

  } // end of each region
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.26.3.23 void runMarketModule ( )

computes st (supply total) and pw (weighted price). Optimisation inside.

Definition at line 226 of file ModelCoreSpatial.cpp.

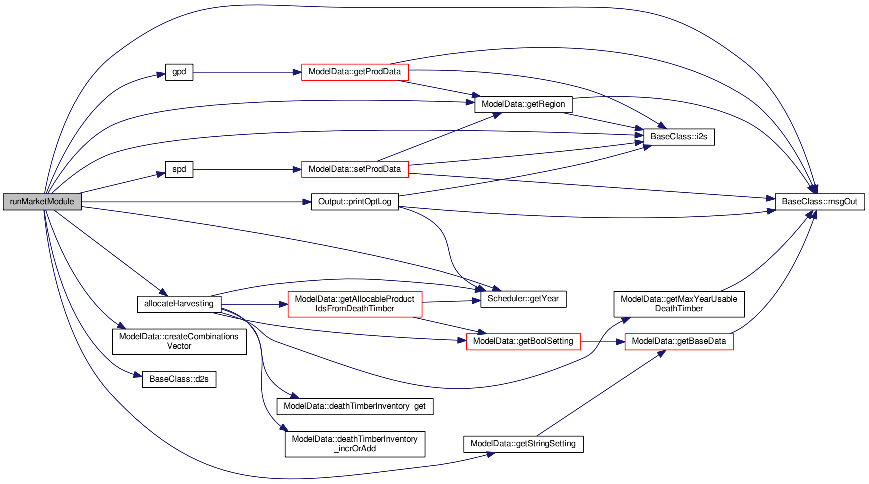
Referenced by runSimulationYear().

```
00226                                 {
00227   msgOut(MSG_INFO, "Starting market module");
00228   static double cumOverHarvesting = 0.0;
00229   int thisYear     =  MTHREAD->SCD->getYear();
00230   int previousYear = MTHREAD->SCD->getYear()-1;
00231 
00232   // *** PRE-OPTIMISATION YEARLY OPERATIONS..
00233   for(uint i=0;i<regIds2.size();i++){
00234     int r2 = regIds2[i];
00235     for(uint sp=0;sp<secProducts.size();sp++){
00236       double g1      = gpd("g1",r2,secProducts[sp],previousYear);
00237       double sigma   = gpd("sigma",r2,secProducts[sp]);
00238       double pc_1    = gpd("pc",r2,secProducts[sp],previousYear);
00239       double dc_1    = gpd("dc",r2,secProducts[sp],previousYear);
00240       double k_1     = gpd("k",r2,secProducts[sp],previousYear);
00241       double sub_d_1 = gpd("sub_d",r2,secProducts[sp],previousYear);
00242 
00243       double k  = (1+g1)*k_1;
00244       double aa = (sigma/(sigma+1))*pc_1*pow(dc_1,-1/sigma);
00245       double gg = dc_1*pow(pc_1+sub_d_1,-sigma); //alpha
00246 
00247       spd(k, "k" ,r2,secProducts[sp]);
00248       spd(aa,"aa",r2,secProducts[sp],DATA_NOW,true);
00249       spd(gg,"gg",r2,secProducts[sp],DATA_NOW,true);
00250     }
00251 
00252     // BB(i,p_pr)   =
       (sigma(p_pr)/(sigma(p_pr)+1))*RPAR('pc',i,p_pr,t-1)*(RPAR('sc',i,p_pr,t-1)**(-1/sigma(p_pr)))*(In(i,p_pr,t-1)/In(i,p_p
00253     // FF(i,p_pr)   =
       RPAR('sc',i,p_pr,t-1)*((RPAR('pc',i,p_pr,t-1))**(-sigma(p_pr)))*(In(i,p_pr,t)/In(i,p_pr,t-1))**(gamma(p_pr)); //chi
00254     for(uint pp=0;pp<priProducts.size();pp++){
00255       double gamma = gpd("gamma",r2,priProducts[pp]); // elast supply to stock
00256       double sigma = gpd("sigma",r2,priProducts[pp]); // elast supply to price
00257       double sigmaCorr = sigma;
00258       double pc_1  = gpd("pc",r2,priProducts[pp],previousYear);
00259       double sc_1  = gpd("sc",r2,priProducts[pp],previousYear);
00260       double in    = gpd("in",r2,priProducts[pp])+gpd("in_deathTimber",r2,
     priProducts[pp]);
00261       double in_1  = gpd("in",r2,priProducts[pp],previousYear)+gpd("in_deathTimber",r2,
     priProducts[pp],previousYear);
00262       double supCorr = 1.0; // Coefficient to reduce supply function when inventory is small
00263       double sub_s_1 = gpd("sub_s",r2,priProducts[pp],previousYear);
00264 
00265       // //When inventary for a resource is almost null and further decreasing supply depends less from the
       price and more from the resource
00266       // No longer needed, but it could be used again if we face a problem where in go to zero due to too
       much harvesting/growth
00267       // //cout << "gamma orig: " << gamma << endl;
00268       // if (in<=0.1 && in <= in_1) { // 0.3
00269       //   gamma = gamma * 1.8; // 1.3: 0.65;
00270       //   sigmaCorr = sigma*0.2; // 0.4
00271       //   //supCorr = 0.7;
00272       //   //cout << "gamma mod: " << gamma << endl;
00273       // }  else if(in<=1.0 && in <= in_1){
00274       //   gamma = gamma * 1.8; // 1.24: 0.62;
00275       //   sigmaCorr = sigma*0.2; // 0.4
00276       //   //supCorr = 0.8;
00277       //   //cout << "gamma mod: " << gamma << endl;
00278       // }
00279 
00280 
00281        //if(in<=5.0){
00282        //  supCorr = 0.8;
00283        //}
00284 
00285 
00286       //double bb = (sigmaCorr/(sigmaCorr+1.0))*pc_1*pow(sc_1,-1.0/sigmaCorr)*pow(in_1/in,gamma/sigmaCorr);
00287       //double ff = sc_1*pow(pc_1,-sigmaCorr)*pow(in/in_1,gamma); //chi
00288       double bb = (sigmaCorr/(sigmaCorr+1.0))*pc_1*pow(sc_1,-1.0/sigmaCorr)*pow(in_1/in,gamma/sigmaCorr)*
     pow(1.0/supCorr,1.0/sigmaCorr);
00289       double ff = sc_1*pow(pc_1+sub_s_1,-sigmaCorr)*pow(in/in_1,gamma)*supCorr; //chi
00290       //double supCorr2 = pow(1.0/supCorr,1.0/sigmaCorr);
00291 
00292       spd(bb,"bb",r2,priProducts[pp],DATA_NOW,true);
00293       spd(ff,"ff",r2,priProducts[pp],DATA_NOW,true);
00294       spd(sigmaCorr,"sigmaCorr",r2,priProducts[pp],DATA_NOW,true);
00295       //spd(supCorr,"supCorr",r2,priProducts[pp],DATA_NOW,true);
00296       //spd(supCorr2,"supCorr2",r2,priProducts[pp],DATA_NOW,true);
00297 
00298     }
00299 
```

```
  } // end for each region in level 2 (and updating variables)



  // *** OPTIMISATION....

  // Create an instance of the IpoptApplication
  //Opt *OPTa = new Opt(MTHREAD);
  //SmartPtr<TNLP> OPTa = new Opt(MTHREAD);
  SmartPtr<IpoptApplication> application = new IpoptApplication();
  string linearSolver = MTHREAD->MD->getStringSetting("linearSolver");
  application->Options()->SetStringValue("linear_solver", linearSolver); // default in ipopt is ma27
  //application->Options()->SetStringValue("hessian_approximation", "limited-memory"); // quasi-newton approximation of the hessian
  //application->Options()->SetIntegerValue("mumps_mem_percent", 100);
  application->Options()->SetNumericValue("obj_scaling_factor", -1); // maximisation
  application->Options()->SetNumericValue("max_cpu_time", 1800); // max 1/2 hour to find the optimus for one single year
  application->Options()->SetStringValue("check_derivatives_for_naninf", "yes");

  // Initialize the IpoptApplication and process the options
  ApplicationReturnStatus status;
  status = application->Initialize();
  if (status != Solve_Succeeded) {
    printf("\n\n*** Error during initialization!\n");
    msgOut(MSG_INFO,"Error during initialization! Do you have the solver compiled for the specified linear solver?");
    return;
  }

  msgOut(MSG_INFO,"Running optimisation problem for this year (it may take a few minutes for large models)..");
  status = application->OptimizeTNLP(MTHREAD->OPT);


  // *** POST OPTIMISATION....

  // post-equilibrium variables->parameters assignments..
  // RPAR(type,i,prd,t)    =  RVAR.l(type,i,prd);
  // EX(i,j,prd,t)         =  EXP.l(i,j,prd);
  // ObjT(t)               =  Obj.l ;
  // ==> in Opt::finalize_solution()

  // Retrieve some statistics about the solve
  if (status == Solve_Succeeded) {
    Index iter_count = application->Statistics()->IterationCount();
    Number final_obj = application->Statistics()->FinalObjective();
    printf("\n*** The problem solved in %d iterations!\n", iter_count);
    printf("\n*** The final value of the objective function is %e.\n", final_obj);
    msgOut(MSG_INFO, "The problem solved successfully in "+i2s(iter_count)+" iterations.");
    int icount = iter_count;
    double obj = final_obj;
    MTHREAD->DO->printOptLog(true, icount, obj);
  } else {
    //Number final_obj = application->Statistics()->FinalObjective();
    cout << "***ERROR: MODEL DIDN’T SOLVE FOR THIS YEAR" << endl;
    msgOut(MSG_CRITICAL_ERROR, "Model DIDN’T SOLVE for this year");
    // IMPORTANT! Don’t place the next two lines above the msgOut() function or it will crash in windows if the user press the stop button
    //Index iter_count = application->Statistics()->IterationCount(); // syserror if model doesn’t solve
    //Number final_obj = application->Statistics()->FinalObjective();
    int icount = 0;
    double obj = 0;
    MTHREAD->DO->printOptLog(false, icount, obj);
  }

  for(uint r2= 0; r2<regIds2.size();r2++){  // you can use r2<=regIds2.size() to try an out-of range memory error that is not detected other than by valgrind (with a message "Invalid read of size 4 in ModelCore::runSimulationYear() in src/ModelCore.cpp:351")
    int regId = regIds2[r2];
    ModelRegion* REG = MTHREAD->MD->getRegion(regId);

    //  // total supply and total demand..
    //  RPAR(’st’,i,prd,t)    =  RPAR(’sl’,i,prd,t)+RPAR(’sa’,i,prd,t);
    //  RPAR(’dt’,i,prd,t)    =  RPAR(’dl’,i,prd,t)+RPAR(’da’,i,prd,t);
    //  // weighted prices.. //changed 20120419
    //  RPAR(’pw’,i,p_tr,t)    = (RPAR(’dl’,i,p_tr,t)*RPAR(’pl’,i,p_tr,t)+RPAR(’da’,i,p_tr,t)*PT(p_tr,t))/RPAR(’dt’,i,p_tr,t) ; //changed 20120419
    //  RPAR(’pw’,i,p_pr,t)    = (RPAR(’sl’,i,p_pr,t)*RPAR(’pl’,i,p_pr,t)+RPAR(’sa’,i,p_pr,t)*PT(p_pr,t))/RPAR(’st’,i,p_pr,t) ; //changed 20120419
    for (uint p=0;p<allProducts.size();p++){
      double st = gpd("sl",regId,allProducts[p])+gpd("sa",regId,allProducts[p]);
      double dt = gpd("dl",regId,allProducts[p])+gpd("da",regId,allProducts[p]);
      spd(st,"st",regId,allProducts[p]);
      spd(st,"st_or",regId,allProducts[p],DATA_NOW,true); // original total supply, not corrected by resetting it to min(st, inv).
      spd(dt,"dt",regId,allProducts[p]);
    }
    for (uint p=0;p<secProducts.size();p++){
      double dl = gpd("dl",regId,secProducts[p]);
      double pl = gpd("pl",regId,secProducts[p]);
      double da = gpd("da",regId,secProducts[p]); // bug corrected 20120913
      double pworld = gpd("pl", WL2,secProducts[p]);
      double dt = gpd("dt",regId,secProducts[p]);
      double pw = dt?(dl*pl+da*pworld)/dt:0.0;
      spd(pw,"pw",regId,secProducts[p]);
    }
    for (uint p=0;p<priProducts.size();p++){
      double sl = gpd("sl",regId,priProducts[p]);
      double pl = gpd("pl",regId,priProducts[p]);
      double sa = gpd("sa",regId,priProducts[p]);  // bug corrected 20120913
      double pworld = gpd("pl", WL2,priProducts[p]);
      double st = gpd("st",regId,priProducts[p]);
      double pw = st?(sl*pl+sa*pworld)/st:0.0;
      spd(pw,"pw",regId,priProducts[p]);
    }

    // Correcting st if this is over the in

    // Create a vector with all possible combinations of primary products
    vector<vector<int>> priPrCombs = MTHREAD->MD->createCombinationsVector(priProducts.size());
    int nPriPrCombs = priPrCombs.size();

    for (uint i=0;i<priPrCombs.size();i++){
      double stMkMod = 0.0;
      double sumIn = REG->inResByAnyCombination[i];
     // double sumIn2 = 0.0;
      for (uint p=0;p<priPrCombs[i].size();p++){
        stMkMod += gpd("st",regId,priProducts[priPrCombs[i][p]]);
        //sumIn2 += gpd("in",regId,priProducts[priPrCombs[i][p]]);
      }

      //if(sumIn<=0.00001){
     //   for (uint p=0;p<priPrCombs[i].size();p++){
     //     spd(0.0,"st",regId,priProducts[priPrCombs[i][p]]);
     //   }
     // } else {
      if(stMkMod>sumIn){ // if we harvested more than available
        string pProductsInvolved = "";
        for (uint p=0;p<priPrCombs[i].size();p++){
          pProductsInvolved += (priProducts[priPrCombs[i][p]]+"; ");
        }
        double inV_over_hV_ratio = stMkMod ? sumIn/stMkMod : 0.0;
        cumOverHarvesting += (stMkMod-sumIn);
        msgOut(MSG_DEBUG, "Overharvesting has happened. Year: "+i2s(thisYear)+ "Region: "+i2s(regId)+"Involved products:  "+pProductsInvolved+". sumIn: "+d2s(sumIn)+" stMkMod:" +d2s(stMkMod) + " cumOverHarvesting: "+d2s(cumOverHarvesting));
        for (uint p=0;p<priPrCombs[i].size();p++){
          double st_orig = gpd("st",regId,priProducts[priPrCombs[i][p]]);
          spd(st_orig*inV_over_hV_ratio,"st",regId,priProducts[priPrCombs[i][p]]);
        }
      }

      //}



    }

    // here we create stFromHarvesting as st - st_from_deathbiomass
    vector <double> total_st(priProducts.size(),0.);
    vector <double> in_deathTimber(priProducts.size(),0.);
    vector <double> in_aliveForest (priProducts.size(),0.);
    for (uint i=0;i<priProducts.size();i++){
      total_st[i] = gpd("st",regId,priProducts[i]);
      in_deathTimber[i] = gpd("in_deathTimber",regId,priProducts[i]);
      in_aliveForest[i] = gpd("in",regId,priProducts[i]);
    }

    vector <double> stFromHarvesting = allocateHarvesting(total_st, regId);

    for (uint i=0;i<priProducts.size();i++){
      spd(stFromHarvesting[i],"stFromHarvesting",regId,priProducts[i],DATA_NOW,true);
    }

  } // end of each region
  if (cumOverHarvesting>0.0){
    msgOut(MSG_DEBUG, "Overharvesting is present. Year: "+i2s(thisYear)+" cumOverHarvesting: "+d2s(cumOverHarvesting));
  }
```

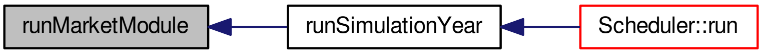
```
00456 }
```

Here is the call graph for this function:



runMarketModule
gpd
ModelData::getProdData
ModelData::getRegion
BaseClass::i2s
spd
ModelData::setProdData
Output::printOptLog
BaseClass::msgOut
allocateHarvesting
ModelData::getAllocableProductIdsFromDeathTimber
Scheduler::getYear
ModelData::getMaxYearUsableDeathTimber
ModelData::createCombinationsVector
ModelData::getBoolSetting
ModelData::getBaseData
BaseClass::d2s
ModelData::deathTimberInventory_get
ModelData::deathTimberInventory_incrOrAdd
ModelData::getStringSetting

Here is the caller graph for this function:



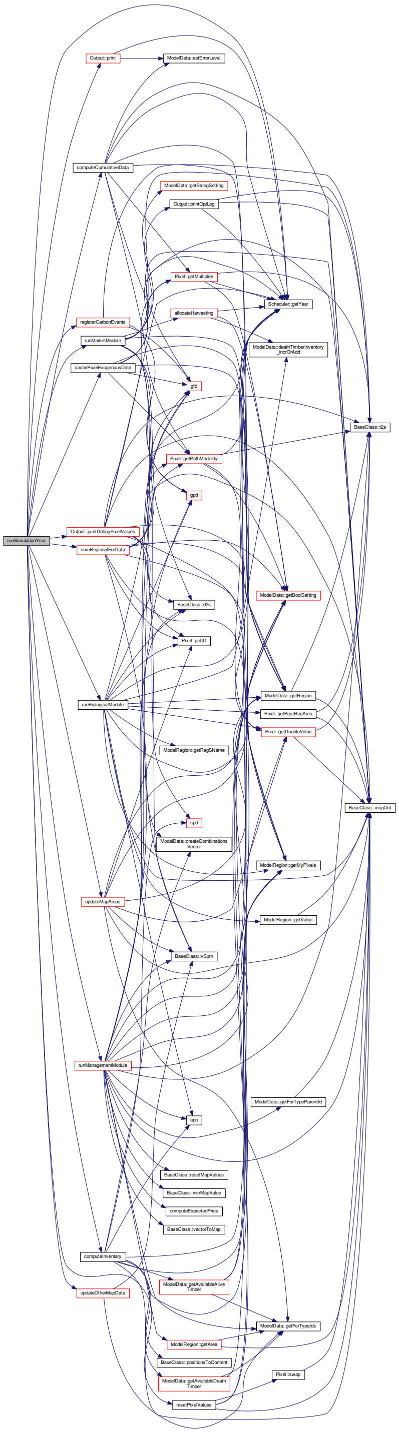
4.26.3.24 void runSimulationYear ( )

Definition at line 75 of file ModelCoreSpatial.cpp.

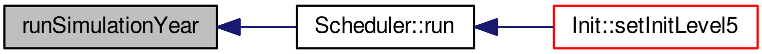
Referenced by Scheduler::run().

```
00075                                    {
00076  int thisYear = MTHREAD->SCD->getYear(); // for debugging
00077  resetPixelValues();           // swap volumes->lagged_volumes and reset the other pixel
vectors
00078  cachePixelExogenousData();    // compute pixel tp, meta and mort
00079  computeInventary();           // in=f(vol_t-1)
00080  runMarketModule();            // RUN THE MARKET OPTIMISATION HERE
00081  computeCumulativeData();      // compute cumTp_exp, vHa_exp
00082  cachePixelExogenousData();
00083  runBiologicalModule();
00084  runManagementModule();
00085  MTHREAD->DO->printDebugPixelValues();
00086  updateMapAreas();
00087  updateOtherMapData();         // update (if the layer exists) other gis-based data, as
volumes and expected returns, taking them from the data in the px object
00088  sumRegionalForData();         // only for printing stats as forest data is never used at
regional level
00089  registerCarbonEvents();
00090  MTHREAD->DO->print();
00091 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



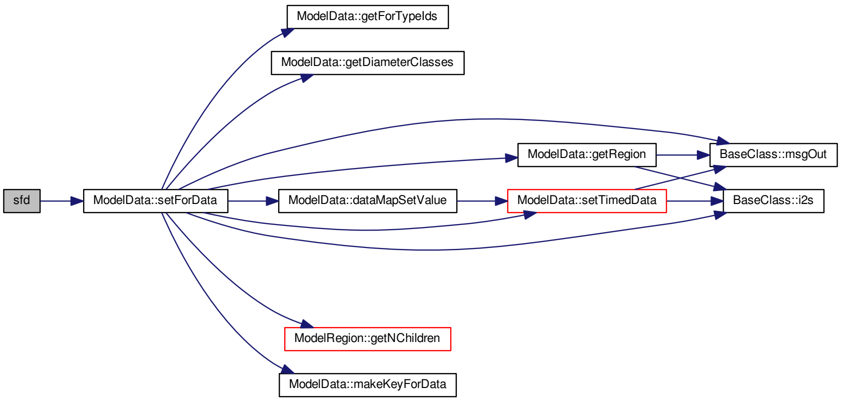
4.26.3.25 void sfd (const double & *value_h,* const string & *type_h,* const int & *regId_h,* const string & *forType_h,* const string & *freeDim_h,* const int & *year* = DATA_NOW*,* const bool & *allowCreate* = `false`) const `[inline]`

Definition at line 119 of file ModelCoreSpatial.h.

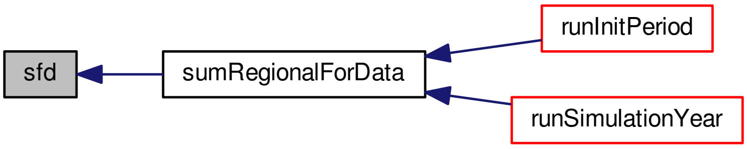
Referenced by sumRegionalForData().

```
00119 {MTHREAD->MD->setForData(value_h, type_h, regId_h, forType_h, freeDim_h, year,
      allowCreate);};
```

Here is the call graph for this function:



Here is the caller graph for this function:



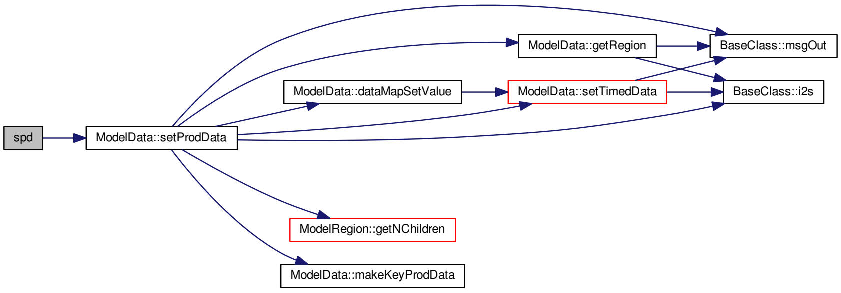
4.26.3.26 void spd (const double & *value_h*, const string & *type_h*, const int & *regId_h*, const string & *prodId_h*, const int & *year* = DATA_NOW, const bool & *allowCreate* = `false`, const string & *freeDim_h* = "") const `[inline]`

Definition at line 118 of file ModelCoreSpatial.h.

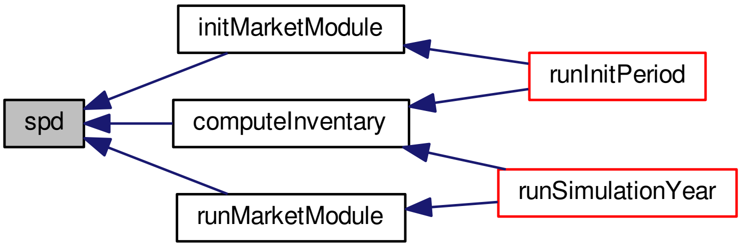
Referenced by computeInventary(), initMarketModule(), and runMarketModule().

```
00118 {MTHREAD->MD->setProdData(value_h, type_h, regId_h, prodId_h, year, allowCreate,
      freeDim_h);};
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.26.3.27 void sumRegionalForData ( )

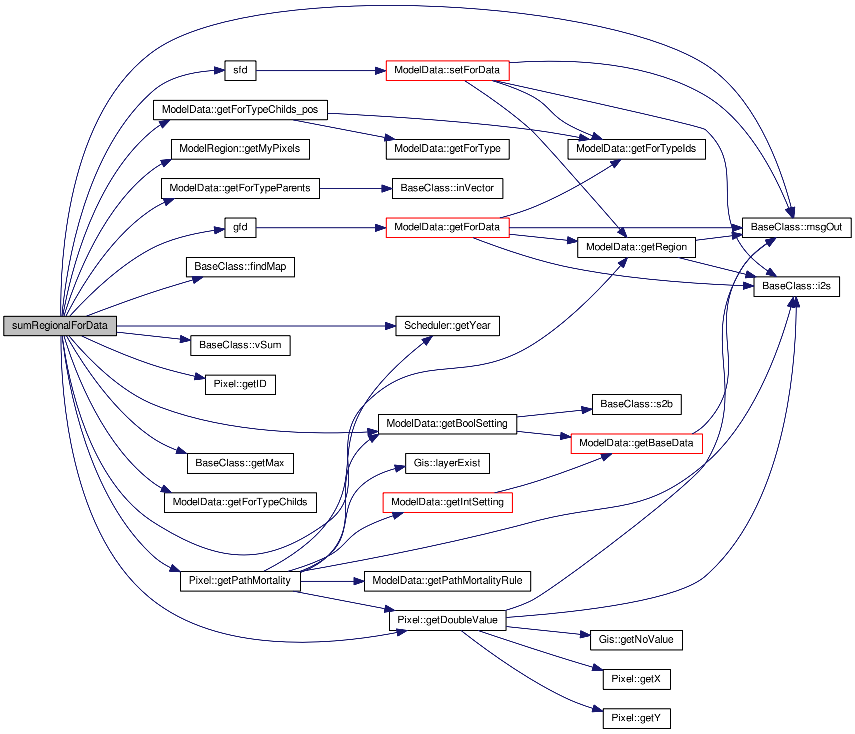
computes vol, hV, harvestedArea, regArea and expReturns at reg level from the pixel level

Definition at line 1782 of file ModelCoreSpatial.cpp.

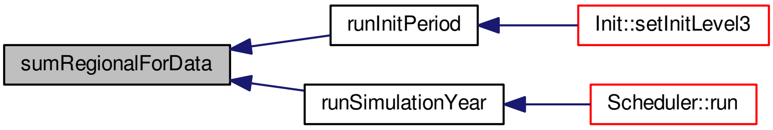
Referenced by runInitPeriod(), and runSimulationYear().

```
                                    {

  msgOut(MSG_INFO, "Summing data pixels->region..");
  //vector <string> outForVariables  = MTHREAD->MD->getStringVectorSetting("outForVariables");
  int currentYear = MTHREAD->SCD->getYear();

  // OLD CODE TO
  for(uint r2= 0; r2<regIds2.size();r2++){
    int regId = regIds2[r2];
    regPx = MTHREAD->MD->getRegion(regId)->getMyPixels();

    for(uint j=0;j<fTypes.size();j++){
      string ft = fTypes[j];

      double regArea = 0.;
      double sumAreaByFt = 0.;
      double pxForAreaByFt = 0.;
      double vReg = 0.;

      for (uint u=0; u<dClasses.size(); u++){
        string dc = dClasses[u];
        double vol =0.;
        double hV = 0.;
        double hArea = 0.;
        double vMort = 0.;
        for (uint p=0;p<regPx.size();p++){
          Pixel* px = regPx[p];
          vol += px->vol.at(j).at(u);
          hV += px->hVol.at(j).at(u);
          hArea += px->hArea.at(j).at(u);
          vMort += px->vMort.at(j).at(u);
        }
        if(u){
          sfd(vol,"vol",regId,ft,dc,DATA_NOW);
          sfd(hV,"hV",regId,ft,dc,DATA_NOW,true);
          sfd(hArea,"harvestedArea",regId,ft,dc,DATA_NOW, true);
          sfd(vMort,"vMort",regId,ft,dc,DATA_NOW,true);
          double vol_1 = gfd("vol",regId,ft,dc,currentYear-1);
          if(vol_1){
            sfd(hV/vol_1,"hr",regId,ft,dc,DATA_NOW, true);
          } else {
            sfd(0.,"hr",regId,ft,dc,DATA_NOW, true);
          }

        }
      }
      for (uint p=0;p<regPx.size();p++){
        Pixel* px = regPx[p];
        vReg += px->vReg.at(j);
        regArea += findMap(px->regArea,currentYear).at(j);
        pxForAreaByFt = (px->getDoubleValue("forArea_"+ft,true)/10000);

        sumAreaByFt += pxForAreaByFt;
        //double debug1 = sumAreaByFt;
        if(! (sumAreaByFt >= 0.0) ){
          msgOut(MSG_CRITICAL_ERROR,"sumAreaByFt is not non negative.");
        }
      }
      sfd(vReg,"vReg",regId,ft,"",DATA_NOW, true);
      sfd(regArea,"regArea",regId,ft,"",DATA_NOW, true);
      sfd(sumAreaByFt,"forArea",regId,ft,"",DATA_NOW, true);
    } // end of for each ft

    for (uint p=0;p<regPx.size();p++){
      Pixel* px = regPx[p];
      double totPxForArea = vSum(px->area);

#ifdef QT_DEBUG
      double totPxForArea_debug = 0.0;
      for(uint j=0;j<fTypes.size();j++){
        string ft = fTypes[j];
        totPxForArea_debug += (px->getDoubleValue("forArea_"+ft,true)/10000);
      }

      if ( (totPxForArea - totPxForArea_debug) > 0.0001 || (totPxForArea - totPxForArea_debug) < -0.0001 ){
        cout << "*** ERROR: area discrepance in pixel " << px->getID() << " of " << (totPxForArea -
totPxForArea_debug) << " ha!" << endl;
        msgOut(MSG_CRITICAL_ERROR,"Total forest area in pixel do not coincide if
 token from layer forArea or (pixel) vector area!");
      }
#endif
    } // end of each pixel

  } // end each region



  // Taking care of expected returns here..
  // (Changed 25/08/2016 afternoon: expRet{ft,r} are now sum{px}{expRet{ft,px}*fArea_{px}}/fArea{r} and no longer sum{px}{expRet{ft,px}*fArea_{px,ft}}/fArea{r,ft} )
  // Also now we report the expReturns by group and by forest, each of which is made only with the best ones within their group

  vector<string>  parentFtypes = MTHREAD->MD->getForTypeParents();

  for(uint r2= 0; r2<regIds2.size();r2++){
    int regId = regIds2[r2];
    regPx = MTHREAD->MD->getRegion(regId)->getMyPixels();
    double totRegionForArea = 0.;
    double totSumExpRet  = 0.;
    vector <double> totSumExpRet_byFTParent(parentFtypes.size(),0.0);
    vector <double> totSumExpRet_byFTypes(fTypes.size(),0.0);

    // First computing the sumExpectedReturns..
    for (uint p=0;p<regPx.size();p++){
      Pixel* px = regPx[p];
      //int debug_pxid = px->getID();
      double pxForArea = vSum(px->area);
      totRegionForArea += pxForArea;
      double bestPxExpectedRet = getMax(px->expectedReturnsNotCorrByRa);
      for(uint i=0;i<parentFtypes.size();i++){
        vector <string> childIds = MTHREAD->MD->getForTypeChilds(parentFtypes[i]);
        vector <int> childPos = MTHREAD->MD->getForTypeChilds_pos(parentFtypes[i]);
        vector<double> pxExpReturnsByChilds(childPos.size(),0.0);
        for(uint j=0;j<childPos.size();j++){
          double pxExpReturn_singleFt = px->expectedReturns.at(childPos[j]);
          // Manual fix to not have the expected returns of ash within the general "broadL" expected returns.
          // To do: remove it after we work on the ash project.. I don't like manual fixes !!!
          pxExpReturnsByChilds.at(j) = (childIds.at(j) == "ash") ? 0.0 : pxExpReturn_singleFt;
          //pxExpReturnsByChilds.at(j) = pxExpReturn_singleFt;
          totSumExpRet_byFTypes.at(childPos[j]) += pxExpReturn_singleFt*pxForArea;
        } // end of each ft
        totSumExpRet_byFTParent[i] += getMax(pxExpReturnsByChilds)*pxForArea;
      } // end for each partentFt
      totSumExpRet += bestPxExpectedRet * pxForArea;
    } // end for each px

    // ..and now computing the expReturns and storing them
    for(uint i=0;i<parentFtypes.size();i++){
      vector <int> childPos = MTHREAD->MD->getForTypeChilds_pos(parentFtypes[i]);
      for(uint j=0;j<childPos.size();j++){
        //double debug1 = totSumExpRet_byFTypes.at(childPos[j])/totRegionForArea;
        sfd(totSumExpRet_byFTypes.at(childPos[j]),"sumExpReturns",regId,fTypes.at(childPos[j]),"",DATA_NOW, true);
        sfd(totSumExpRet_byFTypes.at(childPos[j])/totRegionForArea,"expReturns",regId,fTypes.at(childPos[j]),"",DATA_NOW, true);
      } // end of each ft
      //double debug2 = totSumExpRet_byFTParent.at(i)/totRegionForArea;
      sfd(totSumExpRet_byFTParent.at(i),"sumExpReturns",regId,parentFtypes[i],"",DATA_NOW, true);
      sfd(totSumExpRet_byFTParent.at(i)/totRegionForArea,"expReturns",regId,parentFtypes[i],"",DATA_NOW, true);

    } // end for each partentFt
    //double debug3 = totSumExpRet/totRegionForArea;
    sfd(totSumExpRet,"sumExpReturns",regId,"","",DATA_NOW, true);
    sfd(totSumExpRet/totRegionForArea,"expReturns",regId,"","",DATA_NOW, true);

  } // end for each region

  // Computing pathogens share of forest invasion
  if(MD->getBoolSetting("usePathogenModule")){
      for(uint r2= 0; r2<regIds2.size();r2++){
        int regId = regIds2[r2];
        regPx = MTHREAD->MD->getRegion(regId)->getMyPixels();
        double totalForArea = 0.0;
        double invadedArea  = 0.0;
        for (uint p=0;p<regPx.size();p++){
          Pixel* px = regPx[p];
          int invaded = 0.0;
          for(uint j=0;j<fTypes.size();j++){
            for (uint u=0; u<dClasses.size(); u++){
              if(px->getPathMortality(fTypes[j],dClasses[u]) > 0){
                invaded = 1.0;
              }
            }
          }
          totalForArea += vSum(px->area);
          invadedArea  += vSum(px->area)*invaded;
        }
        sfd(invadedArea/totalForArea,"totalShareInvadedArea",regId,"","",
    DATA_NOW, true);
      }
  } // end we are using path model
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.26.3.28 void updateMapAreas ( )

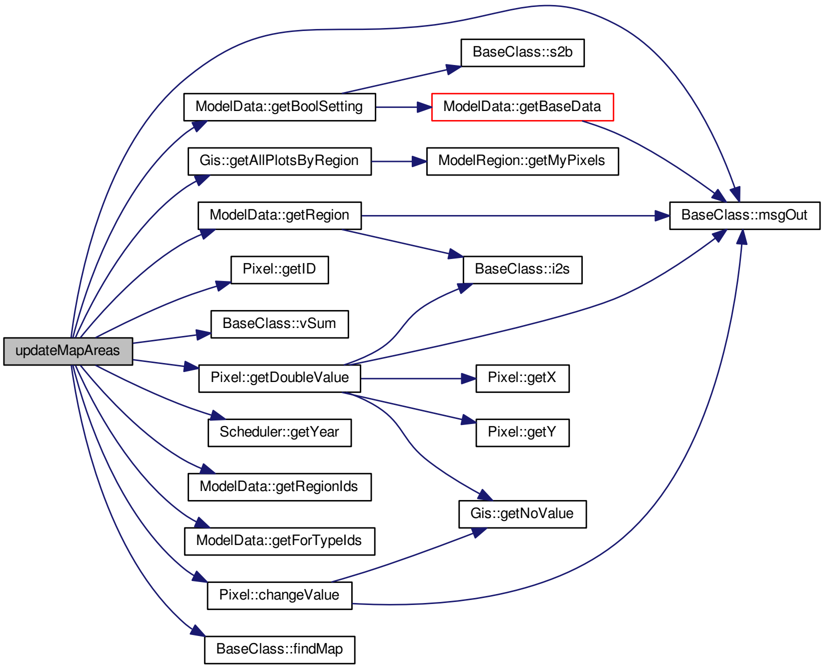
computes forArea_{ft}

Definition at line 1679 of file ModelCoreSpatial.cpp.

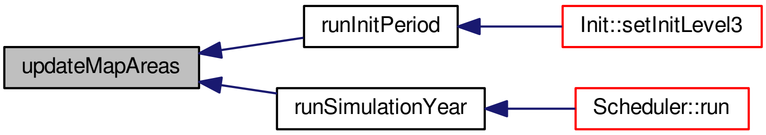
Referenced by runInitPeriod(), and runSimulationYear().

```
                                {
  msgOut(MSG_INFO, "Updating map areas..");

  if (!oldVol2AreaMethod){
    if(!MD->getBoolSetting("usePixelData")) return;
    for(uint i=0;i<regIds2.size();i++){
      ModelRegion* reg = MD->getRegion(regIds2[i]);
      vector <Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(regIds2[i]);
      for (uint p=0;p<rpx.size();p++){
        Pixel* px = rpx[p];
        double pxid= px->getID();
        for(uint j=0;j<fTypes.size();j++){
          string ft = fTypes[j];
          double forArea = vSum(px->area.at(j));
          #ifdef QT_DEBUG
          if(forArea < 0.0 ){
            msgOut(MSG_CRITICAL_ERROR, "Negative forArea in updateMapAreas");
          }
          #endif
          px->changeValue("forArea_"+ft, forArea*10000);
        } // end ft
      } // end px
    } // end region
  } else {
    int currentYear = MTHREAD->SCD->getYear();
    map<int,double> forestArea; // foresta area by each region
    pair<int,double > forestAreaPair;
    vector<int> l2Regions =  MTHREAD->MD->getRegionIds(2, true);
    vector <string> fTypes =  MTHREAD->MD->getForTypeIds();
    int nFTypes = fTypes.size();
    int nL2Regions = l2Regions.size();
    for(int i=0;i<nL2Regions;i++){
      int regId = l2Regions[i];
      vector<Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(regId);
      for(int j=0;j<nFTypes;j++){
        string ft = fTypes[j];
        //double regForArea = reg->getValue("forArea_"+ft);
        //double harvestedArea = gfd("harvestedArea",regId,ft,DIAM_ALL);
        //double regArea = gfd("regArea",regId,ft,DIAM_ALL);
        //cout << "Regid/ft/area/harvested/regeneration: " <<regId<<";"<<ft<<";"<<regForArea<<";"<<harvestedArea<<";" <<regArea<<endl;
        //double newAreaNet = regArea-harvestedArea;
        //double newAreaRatio =  newAreaNet / regForArea;
        for(uint z=0;z<rpx.size();z++){
          Pixel* px = rpx[z];
          double oldValue = px->getDoubleValue("forArea_"+ft,true)/10000;
          double hArea = vSum(px->hArea.at(j));              //bug 20140205 areas in the model are in ha, in the layer in m^2
          double regArea = findMap(px->regArea,currentYear).at(j); //bug 20140205 areas in the model are in ha, in the layer in m^2
          //double newValue = oldValue*(1. + newAreaRatio);
          double newValue = oldValue-hArea+regArea;
          double areaNetOfRegeneration = oldValue-hArea;
          #ifdef QT_DEBUG
          if (areaNetOfRegeneration<0.0){
            msgOut(MSG_CRITICAL_ERROR,"areaNetOfRegeneration negative in updateMapAreas");
          }
          if (newValue<0.0){
            msgOut(MSG_CRITICAL_ERROR,"for area negative in updateMapAreas");
          }
          #endif
          rpx[z]->changeValue("forArea_"+ft, newValue*10000);
        }
      }
    }
  }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.26.3.29 void updateOtherMapData ( )

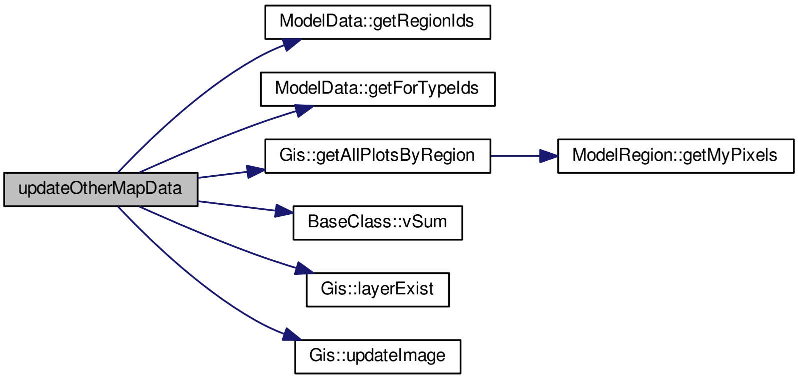
update (if the layer exists) other gis-based data, as volumes and expected returns, taking them from the data in the px object

Definition at line 1745 of file ModelCoreSpatial.cpp.

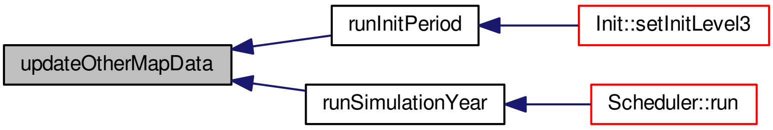
Referenced by runInitPeriod(), and runSimulationYear().

```
01745                                     {
01746
01747 vector<int> l2Regions =  MTHREAD->MD->getRegionIds(2, true);
01748 vector <string> fTypes =  MTHREAD->MD->getForTypeIds();
01749 int nFTypes = fTypes.size();
01750 int nL2Regions = l2Regions.size();
01751 for(int i=0;i<nL2Regions;i++){
  int regId = l2Regions[i];
  vector<Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(regId);
  for(int j=0;j<nFTypes;j++){
    string ft = fTypes[j];
    for(uint z=0;z<rpx.size();z++){
      Pixel* px = rpx[z];
      double vol = vSum(px->vol.at(j));
      double expectedReturns = px->expectedReturns.at(j);
      if(MTHREAD->GIS->layerExist("vol_"+ft)){
        rpx[z]->changeValue("vol_"+ft, vol);
      }
      if(MTHREAD->GIS->layerExist("expectedReturns_"+ft)){
        rpx[z]->changeValue("expectedReturns_"+ft, expectedReturns);
      }
    }
  }
}

// update GUI image..
for(int j=0;j<nFTypes;j++){
  string ft = fTypes[j];
  MTHREAD->GIS->updateImage("vol_"+ft);
  MTHREAD->GIS->updateImage("expectedReturns_"+ft);
}


}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.26.4 Member Data Documentation

4.26.4.1 vector<string> allProducts [private]

Definition at line 131 of file ModelCoreSpatial.h.

Referenced by cacheSettings(), initMarketModule(), registerCarbonEvents(), and runMarketModule().

4.26.4.2 vector< string > dClasses [private]

Definition at line 132 of file ModelCoreSpatial.h.

Referenced by allocateHarvesting(), cachePixelExogenousData(), cacheSettings(), computeCumulativeData(), computeInventary(), initialiseDeathTimber(), initializePixelArea(), initializePixelVolumes(), loadExogenousForestLayers(), runBiologicalModule(), runManagementModule(), and sumRegionalForData().

4.26.4.3 int firstYear [private]

Definition at line 124 of file ModelCoreSpatial.h.

Referenced by cachePixelExogenousData(), cacheSettings(), computeCumulativeData(), initializePixelVolumes(), and initMarketModule().

4.26.4.4 string forestAreaChangeMethod [private]

Definition at line 142 of file ModelCoreSpatial.h.

Referenced by cacheSettings(), and runManagementModule().

4.26.4.5 vector< string > fTypes [private]

Definition at line 134 of file ModelCoreSpatial.h.

Referenced by allocateHarvesting(), assignSpMultiplierPropToVols(), cachePixelExogenousData(), cacheSettings(), computeCumulativeData(), computeInventary(), initialiseCarbonModule(), initialiseDeathTimber(), initializePixelArea(), initializePixelVolumes(), loadExogenousForestLayers(), registerCarbonEvents(), runBiologicalModule(), runManagementModule(), sumRegionalForData(), updateMapAreas(), and updateOtherMapData().

4.26.4.6 double ir [private]

Definition at line 143 of file ModelCoreSpatial.h.

Referenced by cacheSettings(), loadExogenousForestLayers(), and runManagementModule().

4.26.4.7 vector< vector < int > > l2r [private]

Definition at line 135 of file ModelCoreSpatial.h.

Referenced by cacheSettings(), initMarketModule(), printDebugInitRegionalValues(), and registerCarbonEvents().

4.26.4.8 ModelData* MD [private]

Definition at line 120 of file ModelCoreSpatial.h.

Referenced by allocateHarvesting(), assignSpMultiplierPropToVols(), cacheSettings(), computeCumulativeData(), computeInventary(), initialiseDeathTimber(), initializePixelArea(), initializePixelVolumes(), loadExogenousForestLayers(), runBiologicalModule(), runManagementModule(), sumRegionalForData(), and updateMapAreas().

4.26.4.9 string natRegAllocation [private]

Definition at line 137 of file ModelCoreSpatial.h.

Referenced by cacheSettings(), and runManagementModule().

4.26.4.10 bool oldVol2AreaMethod [private]

Definition at line 141 of file ModelCoreSpatial.h.

Referenced by cacheSettings(), runBiologicalModule(), and updateMapAreas().

4.26.4.11 vector<string> pDClasses [private]

Definition at line 133 of file ModelCoreSpatial.h.

Referenced by cacheSettings().

4.26.4.12 vector<string> priProducts [private]

Definition at line 129 of file ModelCoreSpatial.h.

Referenced by allocateHarvesting(), cacheSettings(), computeInventary(), initialiseCarbonModule(), initMarketModule(), printDebugInitRegionalValues(), registerCarbonEvents(), runBiologicalModule(), runManagementModule(), and runMarketModule().

4.26.4.13 vector<int> regIds2 [private]

Definition at line 128 of file ModelCoreSpatial.h.

Referenced by assignSpMultiplierPropToVols(), cachePixelExogenousData(), cacheSettings(), computeCumulativeData(), computeInventary(), initialiseCarbonModule(), initialiseDeathTimber(), initializePixelArea(), initializePixelVolumes(), initMarketModule(), loadExogenousForestLayers(), registerCarbonEvents(), resetPixelValues(), runBiologicalModule(), runManagementModule(), runMarketModule(), sumRegionalForData(), and updateMapAreas().

4.26.4.14 vector<Pixel*> regPx [private]

Definition at line 139 of file ModelCoreSpatial.h.

Referenced by cachePixelExogenousData(), computeCumulativeData(), computeInventary(), loadExogenousForestLayers(), resetPixelValues(), runBiologicalModule(), runManagementModule(), and sumRegionalForData().

4.26.4.15 string regType [private]

Definition at line 136 of file ModelCoreSpatial.h.

Referenced by cacheSettings(), and runBiologicalModule().

4.26.4.16 bool rescaleFrequencies [private]

Definition at line 140 of file ModelCoreSpatial.h.

Referenced by cacheSettings(), and runManagementModule().

4.26.4.17 int secondYear `[private]`

Definition at line 125 of file ModelCoreSpatial.h.

Referenced by cacheSettings(), initMarketModule(), printDebugInitRegionalValues(), and runBiologicalModule().

4.26.4.18 vector<string> secProducts `[private]`

Definition at line 130 of file ModelCoreSpatial.h.

Referenced by cacheSettings(), initialiseCarbonModule(), initMarketModule(), registerCarbonEvents(), and runMarketModule().

4.26.4.19 int thirdYear `[private]`

Definition at line 126 of file ModelCoreSpatial.h.

Referenced by cacheSettings().

4.26.4.20 int WL2 `[private]`

Definition at line 127 of file ModelCoreSpatial.h.

Referenced by cacheSettings(), initMarketModule(), runManagementModule(), and runMarketModule().

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

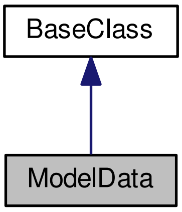
- /home/lobianco/git/ffsm_pp/src/ModelCoreSpatial.h
- /home/lobianco/git/ffsm_pp/src/ModelCoreSpatial.cpp

4.27 ModelData Class Reference

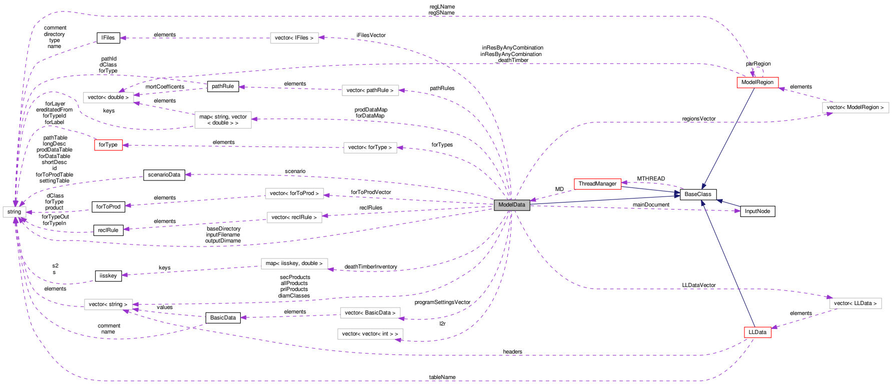
Regional data, including macros and settings.

`#include <ModelData.h>`

Inheritance diagram for ModelData:



Collaboration diagram for ModelData:



Public Member Functions

- ModelData (ThreadManager ∗MTHREAD_h)
- ∼ModelData ()
- void loadInput ()

  Unzip the OpenOffice input file (NEW 2008.05.13)
- void loadDataFromCache (string tablename)

  Load data from a cached CSV instead of the openoffice file.
- vector< string > getScenarios ()
- int getScenarioIndex ()
- bool delDir (QString dirname)

  Recursivelly delete a directory.
- void setScenarioData ()

  Set the infos about this scenario (long description and overriding tables)
- void setDefaultSettings ()
- void setScenarioSettings ()
- void createRegions ()
- void setDefaultForData ()
- void setScenarioForData ()
- void setDefaultProdData ()
- void setScenarioProdData ()
- void setDefaultProductResourceMatrixLink ()
- void setScenarioProductResourceMatrixLink ()
- void setForestTypes ()
- void setReclassificationRules ()
- void setDefaultPathogenRules ()
- void setScenarioPathogenRules ()
- void applyOverrides ()

  Cancel all reg1 level data and trasform them in reg2 level if not already existing.
- void applyDebugMode ()

  Works only a specified subset of regions and products.
- void setSpace ()
- string getOutputDirectory () const

  Return a vector of objects that together provide the specified resource in the specified quantity.
- int getFilenamesByDir (const string &dir, vector< string > &files, const string &filter="")

  Return a list of files in a directory.

- string getFilenameByType (string type_h)
- LLData getTable (string tableName_h, int debugLevel=MSG_CRITICAL_ERROR)
- vector< IFiles > getIFilesVector () const
- string getBaseDirectory () const
- ModelRegion ∗ getRegion (int regId_h)
- bool regionExist (const int &regId_h) const
- vector< ModelRegion ∗ > getAllRegions (bool excludeResidual=true)
- vector< int > getRegionIds (int level_h, bool excludeResidual=true)
- vector< vector< int > > getRegionIds (bool excludeResidual=true)
- string regId2RegSName (const int &regId_h) const
- int regSName2RegId (const string &regSName_h) const
- int getNForTypes ()
- int getNReclRules ()
- forType ∗ getForType (int position)
- forType ∗ getForType (string &forTypeId_h)
- int getForTypeCounter (string &forTypeId_h, bool all=false)

  By default it doesn't return forTypes used only as input.
- vector< string > getForTypeIds (bool all=false)

  By default it doesn't return forTypes used only as input.
- string getForTypeParentId (const string &forTypeId_h)
- vector< string > getForTypeChilds (const string &forTypeId_h)
- vector< int > getForTypeChilds_pos (const string &forTypeId_h, bool all=false)
- vector< string > getForTypeParents ()
- int getNForTypesChilds (const string &forTypeId_h)
- reclRule ∗ getReclRule (int position)
- vector< string > getDiameterClasses (bool productionOnly=false)
- const bool assessProdPossibility (const string &prod_h, const string &forType_h, const string &dClass_h)

  A simple function to assess if a specified product can be made by a certain forest type and diameter class.
- const int getMaxYearUsableDeathTimber (const string &prod_h, const string &forType_h, const string &d←Class_h)
- const int getMaxYearUsableDeathTimber ()
- int setErrorLevel (int errorLevel_h)
- bool getTempBool ()
- vector< vector< int > > createCombinationsVector (const int &nItems)

  Return a vector containing any possible combination of nItems items (including any possible subset). The returned vector has in each slot the items present in that specific combination.
- double getTimedData (const vector< double > &dated_vector, const int &year_h) const

  Return the value for the specified year in a timelly ordered vector, taking the last value if this is smaller than the required position.
- void setTimedData (const double &value_h, vector< double > &dated_vector, const int &year_h, const int &MSG_LEVEL=MSG_WARNING)
- int getIntSetting (const string &name_h, int position=0) const
- double getDoubleSetting (const string &name_h, int position=0) const
- string getStringSetting (const string &name_h, int position=0) const
- bool getBoolSetting (const string &name_h, int position=0) const
- vector< int > getIntVectorSetting (const string &name_h) const
- vector< double > getDoubleVectorSetting (const string &name_h) const
- vector< string > getStringVectorSetting (const string &name_h) const
- vector< bool > getBoolVectorSetting (const string &name_h) const
- const double getProdData (const string &type_h, const int &regId_h, const string &prodId_h, const int &year=DATA_NOW, const string &freeDim_h="")
- const double getForData (const string &type_h, const int &regId_h, const string &forType_h, const string &freeDim_h, const int &year=DATA_NOW)

- void setProdData (const double &value_h, const string &type_h, const int &regId_h, const string &prodId_h, const int &year=DATA_NOW, const bool &allowCreate=false, const string &freeDim_h="")
- void setForData (const double &value_h, const string &type_h, const int &regId_h, const string &forType_h, const string &freeDim_h, const int &year=DATA_NOW, const bool &allowCreate=false)
- string makeKeyProdData (const string &parName, const string &regId, const string &prod, const string &freeDim="") const
- string makeKeyForData (const string &parName, const string &regId, const string &forType, const string &diamClass) const
- void unpackKeyProdData (const string &key, string &parName, int &regId, string &prod, string &freeDim) const
- void unpackKeyForData (const string &key, string &parName, int &regId, string &forType, string &diamClass) const
- vector$<$ pathRule $*$ $>$ getPathMortalityRule (const string &forType, const string &dC)

  Return the pathogen mortality rule(s) associated with a given ft and dc (plural as more than a single pathogen could be found)
- double calculateAnnualisedEquivalent (double amount_h, int years_h)

  Calculate the annual equivalent flow.
- double calculateAnnualisedEquivalent (double amount_h, double years_h)

  Transform the double to the highest integer and call calculateAnnualisedEquivalent(double amount_h, int years_h)
- void setOutputDirectory (const char $*$output_dirname_h)
- void setBaseDiretory (string baseDirectory_h)
- void addSetting (string name_h, vector$<$ string $>$ values_h, int type_h, string comment_h)
- void addSetting (string name_h, string value_h, int type_h, string comment_h)
- void cacheSettings ()

  Called after input reading, it fix frequently used data;.
- int getCachedInitialYear ()
- void setBasicData (const string &name_h, int value, int position=0)
- void setBasicData (const string &name_h, double value, int position=0)
- void setBasicData (const string &name_h, string value, int position=0)
- void setBasicData (const string &name_h, bool value, int position=0)
- void deathTimberInventory_incrOrAdd (const iisskey &thekey, double value_h)
- void deathTimberInventory_incr (const iisskey &thekey, double value_h)
- double deathTimberInventory_get (const iisskey &thekey)
- map$<$ iisskey, double $>$ $*$ getDeathTimberInventory ()
- double getAvailableDeathTimber (const vector$<$ string $>$ &primProd_h, int regID_h, int year_h)

  Returns the timber available for a given set of primary products as stored in the deathTimberInventory map.
- double getAvailableAliveTimber (const vector$<$ string $>$ &primProd_h, int regId_h)

  Returns the timber available for a given set of primary products as stored in the px->vol_l vector.
- vector$<$ int $>$ getAllocableProductIdsFromDeathTimber (const int &regId_h, const string &ft, const string &dc, const int &harvesting_year, int request_year=DATA_NOW)

  Returns the ids of the primary products that is possible to obtain using the timber recorded death in the specific year, ft, dc combination.

Public Attributes

- scenarioData scenario

Private Member Functions

- string getBaseData (const string &name_h, int type_h, int position=0)
- vector< string > getVectorBaseData (const string &name_h, int type_h)
- void setBasicData (const string &name_h, string value, int type_h, int position)
- bool dataMapCheckExist (const DataMap &map, const string &search_for, const bool &exactMatch=true) const
- double dataMapGetValue (const DataMap &map, const string &search_for, const int &year_h, const bool &exactMatch=true)
- int dataMapSetValue (DataMap &map, const string &search_for, const double &value_h, const int &year_h, const bool &exactMatch=true)

Private Attributes

- string inputFilename
- string outputDirname
- string baseDirectory
- map< string, vector< double > > forDataMap

  Forestry data.
- map< string, vector< double > > prodDataMap

  Product data.
- vector< forToProd > forToProdVector

  Vector of coefficients from forest resources to primary products.
- vector< IFiles > iFilesVector

  List of all input files. Simple (struct)
- vector< BasicData > programSettingsVector

  Setting data. Simple (struct)
- vector< LLData > LLDataVector

  Vector of Low Level Data.
- vector< ModelRegion > regionsVector

  Vector of modelled regions.
- vector< forType > forTypes

  Vector of forest types.
- vector< reclRule > reclRules

  Vector of reclassification rules.
- vector< pathRule > pathRules

  Vector of pathogen rules.
- vector< vector< int > > l2r

  Region2 ids.
- map< iisskey, double > deathTimberInventory

  Map that register the death of biomass still usable as timber by year, l2_region, forest type and diameter class [Mm^3 wood].
- vector< string > diamClasses

  Diameter classes.
- int cached_initialYear
- vector< string > priProducts
- vector< string > secProducts
- vector< string > allProducts
- bool tempBool

  a temporary bool variable used for various functions
- InputNode mainDocument

  For each agricultural soil type (as defined in the setting "agrLandTypes") this list define the objects that can be placed on that soil type.
- int errorLevel

Friends

- void Output::printForestData (bool finalFlush=false)
- void Output::printProductData (bool finalFlush=false)

Additional Inherited Members

4.27.1 Detailed Description

Regional data, including macros and settings.

All regional data are within this class. It may have linked other data-classes.
On some variables ModelData has just the definition of the objects, but the values may change at the agent-level. This is why each agent has a "personal copy" of them.

Author

Antonello Lobianco

Definition at line 79 of file ModelData.h.

4.27.2 Constructor & Destructor Documentation

4.27.2.1 ModelData (ThreadManager * *MTHREAD_h*)

Definition at line 61 of file ModelData.cpp.

```
00061                                             {
00062   MTHREAD = MTHREAD_h;
00063   errorLevel = MSG_ERROR;
00064 }
```

4.27.2.2 ∼ModelData ( )

Definition at line 66 of file ModelData.cpp.

```
00066                         {
00067
00068 }
```

4.27.3 Member Function Documentation

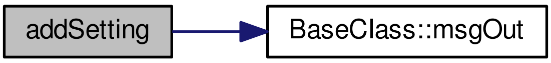
4.27.3.1 void addSetting (string *name_h*, vector< string > *values_h*, int *type_h*, string *comment_h*)

Definition at line 253 of file ModelData.cpp.

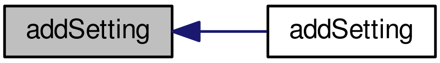
Referenced by addSetting().

```
00253                                                                                              {
00254
00255   for (uint i=0;i<programSettingsVector.size();i++){
00256     if (programSettingsVector.at(i).name == name_h){
00257       msgOut(MSG_ERROR, "I already have setting "+name_h+".. Nothing is added..");
00258       return;
00259     }
00260   }
00261   BasicData SETT;
00262   SETT.name = name_h;
00263   SETT.values = values_h;
00264   SETT.type= type_h;
00265   SETT.comment = comment_h;
00266   programSettingsVector.push_back(SETT);
00267 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

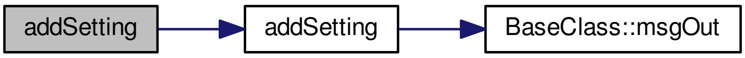


4.27.3.2 void addSetting (string *name_h*, string *value_h*, int *type_h*, string *comment_h*)

Definition at line 270 of file ModelData.cpp.

```
00270                                                                                    {
00271   vector <string> values;
00272   values.push_back(value_h);
00273   addSetting(name_h, values, type_h, comment_h);
00274 }
```

Here is the call graph for this function:



4.27.3.3 void applyDebugMode ( )

Works only a specified subset of regions and products.

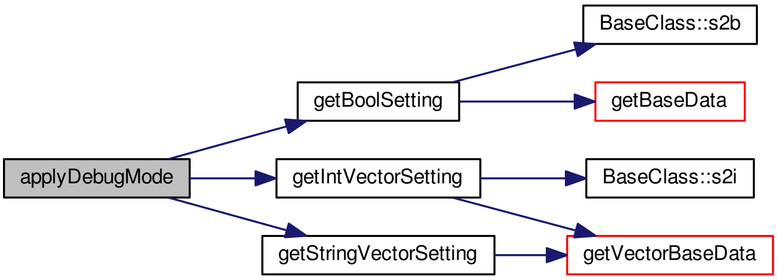
The applyDebugMode flag all level2 regions not in the "debugRegions" option as "residual" (so they are in the map but not in the model code) and remove the primary and secondary products that are not included in the debugPriProducts and debugSecProducts options.

Definition at line 910 of file ModelData.cpp.

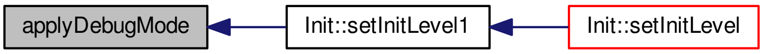
Referenced by Init::setInitLevel1().

```
00910                            {
00911   if(! getBoolSetting("debugFlag")) return;
00912
00913   vector <int> debugRegions = getIntVectorSetting("debugRegions");
00914   vector <string> debugPriProducts = getStringVectorSetting("debugPriProducts");
00915   vector <string> debugSecProducts = getStringVectorSetting("debugSecProducts");
00916
00917   for(uint i=0;i< regionsVector.size();i++){
00918     if (regionsVector[i].getRegLevel()==2){
00919       bool found= false;
00920       for(uint j=0;j<debugRegions.size();j++){
00921         if (debugRegions[j] == regionsVector[i].getRegId()){
00922           found = true;
00923           break;
00924         }
00925       }
00926       if(!found){ // not in the list to keep
00927         regionsVector[i].setIsResidual(true);
00928       }
00929     }
00930   }
00931
00932   for (uint i=0; i<programSettingsVector.size();i++){
00933     if (programSettingsVector.at(i).name == "priProducts"){
00934       programSettingsVector.at(i).values = debugPriProducts;
00935     } else if (programSettingsVector.at(i).name == "secProducts"){
00936       programSettingsVector.at(i).values = debugSecProducts;
00937     }
00938   }
00939
00940 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.3.4 void applyOverrides ( )

Cancel all reg1 level data and trasform them in reg2 level if not already existing.

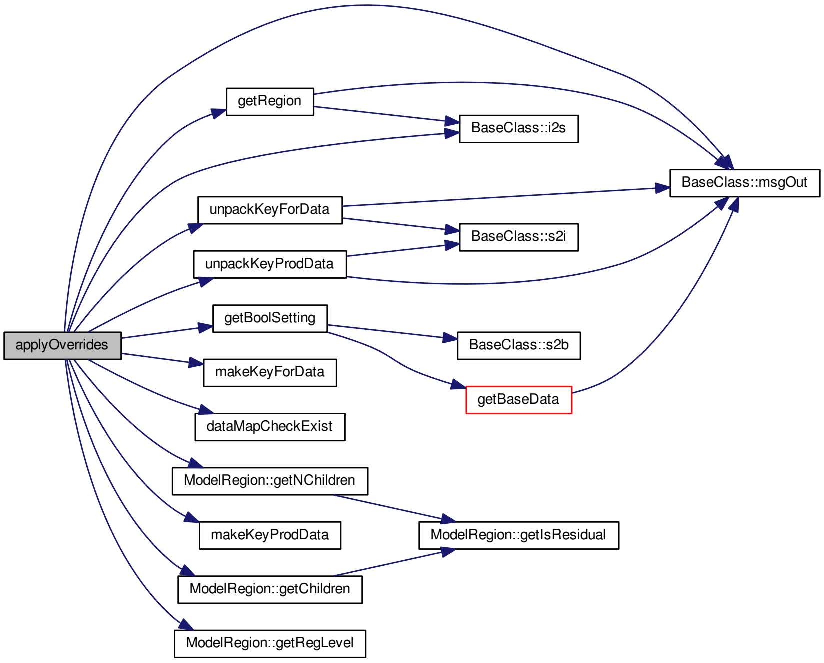
Definition at line 720 of file ModelData.cpp.

Referenced by Init::setInitLevel1().

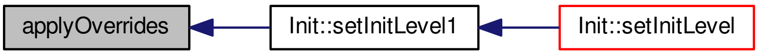
```
00720                          {
00721
00722   if(!getBoolSetting("applyOverriding")) return;
00723   msgOut(MSG_INFO, "Starting regional overriding analysis..");
00724
00725   DataMap::iterator p;
00726   string parName,prod,freeDim,forType,diamClass, key;
00727   int regId;
00728   DataMap toBeAdded;
00729   vector <string> keysToRemove;
00730
00731
00732   //apply override from level 0 to level 1 for forestry data
00733   toBeAdded.clear();
00734   keysToRemove.clear();
00735   for(p=forDataMap.begin();p!=forDataMap.end();p++){
00736     unpackKeyForData(p->first,parName,regId,forType,diamClass);
00737     //if(!regionExist(regId)) continue;
00738     if(getRegion(regId)->getRegLevel() == 0){
00739       vector<ModelRegion*> childs = getRegion(regId)->getChildren(false);
00740       for(uint j=0;j<childs.size();j++){
00741         bool found = false;
00742         key = makeKeyForData(parName,i2s(childs[j]->getRegId()),forType,diamClass);
00743         if (!dataMapCheckExist(forDataMap,key,true)){
00744           toBeAdded.insert(DataPair(key,p->second));
00745         }
00746       }
00747       keysToRemove.push_back(p->first);
```

```
    }
  }
  forDataMap.insert(toBeAdded.begin(),toBeAdded.end());
  for(uint i=0;i<keysToRemove.size();i++){
    DataMap::iterator rem = forDataMap.find(keysToRemove[i]);
    if(rem != forDataMap.end()){
      forDataMap.erase(rem);
    }
  }




  //apply override from level 1 to level 2 for forestry data
  toBeAdded.clear();
  keysToRemove.clear();
  for(p=forDataMap.begin();p!=forDataMap.end();p++){
    unpackKeyForData(p->first,parName,regId,forType,diamClass);
    //if(!regionExist(regId)) continue;
    if(getRegion(regId)->getRegLevel() == 1){
      vector<ModelRegion*> childs = getRegion(regId)->getChildren(false);
      for(uint j=0;j<childs.size();j++){
        bool found = false;
        key = makeKeyForData(parName,i2s(childs[j]->getRegId()),forType,diamClass);
        if (!dataMapCheckExist(forDataMap,key,true)){
          toBeAdded.insert(DataPair(key,p->second));
        }
      }
      keysToRemove.push_back(p->first);
    }
  }
  forDataMap.insert(toBeAdded.begin(),toBeAdded.end());
  for(uint i=0;i<keysToRemove.size();i++){
    DataMap::iterator rem = forDataMap.find(keysToRemove[i]);
    if(rem != forDataMap.end()){
      forDataMap.erase(rem);
    }
  }

  //apply override from level 0 to level 1 for production data
  toBeAdded.clear();
  keysToRemove.clear();
  for(p=prodDataMap.begin();p!=prodDataMap.end();p++){
    unpackKeyProdData(p->first,parName,regId,prod,freeDim);
    //if(!regionExist(regId)) continue;
    if(getRegion(regId)->getRegLevel() == 0){
      vector<ModelRegion*> childs = getRegion(regId)->getChildren(false);
      for(uint j=0;j<childs.size();j++){
        bool found = false;
        key = makeKeyProdData(parName,i2s(childs[j]->getRegId()),prod,freeDim);
        if (!dataMapCheckExist(prodDataMap,key,true)){
          toBeAdded.insert(DataPair(key,p->second));
        }
      }
      //prodDataMap.erase(p);
      //p--;
      keysToRemove.push_back(p->first);
    }
  }
  prodDataMap.insert(toBeAdded.begin(),toBeAdded.end());
  for(uint i=0;i<keysToRemove.size();i++){
    DataMap::iterator rem = prodDataMap.find(keysToRemove[i]);
    if(rem != prodDataMap.end()){
      prodDataMap.erase(rem);
    }
  }


  //apply override from level 1 to level 2 for production data
  toBeAdded.clear();
  keysToRemove.clear();
  for(p=prodDataMap.begin();p!=prodDataMap.end();p++){
    string debug = p->first;
    unpackKeyProdData(p->first,parName,regId,prod,freeDim);
    //if(!regionExist(regId)) continue;
    if(getRegion(regId)->getRegLevel() == 1){
      vector<ModelRegion*> childs = getRegion(regId)->getChildren(false);
      for(uint j=0;j<childs.size();j++){
        bool found = false;
        key = makeKeyProdData(parName,i2s(childs[j]->getRegId()),prod,freeDim);
        if (!dataMapCheckExist(prodDataMap,key,true)){
          toBeAdded.insert(DataPair(key,p->second));
        }
      }
      //prodDataMap.erase(p);
      //p--;
      keysToRemove.push_back(p->first);
    }
  }
  prodDataMap.insert(toBeAdded.begin(),toBeAdded.end());
    for(uint i=0;i<keysToRemove.size();i++){
    DataMap::iterator rem = prodDataMap.find(keysToRemove[i]);
    if(rem != prodDataMap.end()){
      prodDataMap.erase(rem);
    }
  }

  //apply override from level 0 to level 1 for reclassification rules
  for(uint i=0;i<reclRules.size();i++){
    if(reclRules[i].regId == 0){
      //if(!regionExist(reclRules[i].regId)) continue;
      for(uint j=0;j<getRegion(reclRules[i].regId)->getNChildren(false);j++){
        vector<ModelRegion*> childs = getRegion(reclRules[i].regId)->getChildren(false);
        bool found = 0;
        for(uint z=0;z<reclRules.size();z++){
          if(   reclRules[z].regId == childs[j]->getRegId()
             && reclRules[z].forTypeIn == reclRules[i].forTypeIn
             && reclRules[z].forTypeOut == reclRules[i].forTypeOut
          ){
            found = true; // do nothing, this child has been already manually overritten
            break;
          }
        }
        if(!found){
          reclRule RR;
          RR.regId      = childs[j]->getRegId();
          RR.forTypeIn  = reclRules[i].forTypeIn;
          RR.forTypeOut = reclRules[i].forTypeOut;
          RR.coeff      = reclRules[i].coeff;
          reclRules.push_back(RR);
        }
      }
      reclRules.erase(reclRules.begin()+i);
      i--;
    }
  }

  //apply override from level 1 to level 2 for reclassification rules
  for(uint i=0;i<reclRules.size();i++){
    //if(!regionExist(reclRules[i].regId)) continue;
    if(getRegion(reclRules[i].regId)->getRegLevel() == 1){
      for(uint j=0;j<getRegion(reclRules[i].regId)->getNChildren(false);j++){
        vector<ModelRegion*> childs = getRegion(reclRules[i].regId)->getChildren(false);
        bool found = 0;
        for(uint z=0;z<reclRules.size();z++){
          if(   reclRules[z].regId == childs[j]->getRegId()
             && reclRules[z].forTypeIn == reclRules[i].forTypeIn
             && reclRules[z].forTypeOut == reclRules[i].forTypeOut
          ){
            found = true; // do nothing, this child has been already manually overritten
            break;
          }
        }
        if(!found){
          reclRule RR;
          RR.regId      = childs[j]->getRegId();
          RR.forTypeIn  = reclRules[i].forTypeIn;
          RR.forTypeOut = reclRules[i].forTypeOut;
          RR.coeff      = reclRules[i].coeff;
          reclRules.push_back(RR);
        }
      }
      reclRules.erase(reclRules.begin()+i);
      i--;
    }
  }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.3.5 const bool assessProdPossibility (const string & *prod_h*, const string & *forType_h*, const string & *dClass_h*)

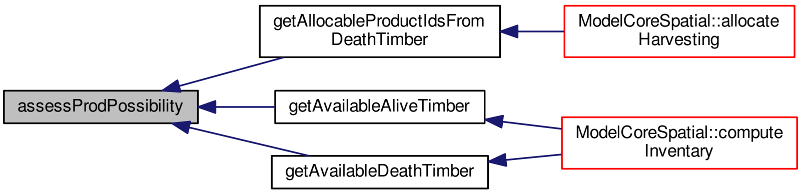
A simple function to assess if a specified product can be made by a certain forest type and diameter class.

Definition at line 413 of file ModelData.cpp.

Referenced by getAllocableProductIdsFromDeathTimber(), getAvailableAliveTimber(), and getAvailableDeathTimber().

```
00413                                                                                                      {
00414   bool ok=false;
00415   for(uint i=0;i<forToProdVector.size();i++){
00416     if(    forToProdVector[i].product == prod_h
00417       && forToProdVector[i].forType == forType_h
00418       && forToProdVector[i].dClass  == dClass_h
00419     ){
00420       return true;
00421     }
00422   }
00423   return false;
00424 }
```

Here is the caller graph for this function:


getAllocableProductIdsFrom DeathTimber
ModelCoreSpatial::allocate Harvesting
assessProdPossibility
getAvailableAliveTimber
ModelCoreSpatial::compute Inventary
getAvailableDeathTimber

4.27.3.6 void cacheSettings ( )

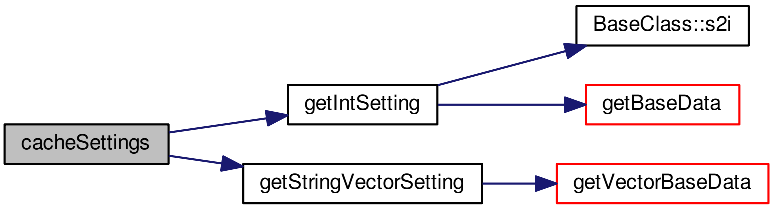
Called after input reading, it fix frequently used data;.

Definition at line 277 of file ModelData.cpp.

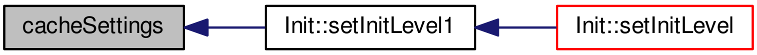
Referenced by Init::setInitLevel1().

```
00277                        {
00278   cached_initialYear = getIntSetting("initialYear");
00279   diamClasses = getStringVectorSetting("dClasses");
00280   priProducts = getStringVectorSetting("priProducts");
00281   secProducts = getStringVectorSetting("secProducts");
00282   allProducts = priProducts;
00283   allProducts.insert( allProducts.end(), secProducts.begin(),
      secProducts.end() );
00284 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.3.7 double calculateAnnualisedEquivalent (double *amount_h,* int *years_h*)

Calculate the annual equivalent flow.

calculating the discount factor

Revenues at years n will be transforemed as average year rate as

av.y.rev = rev(n)/ ((1+ir)^(n-1)+(1+ir)^(n-2)+(1+ir)^(n-3)+...+(1+ir)^(n-n))

Objective is to have the present value of the final harvest (A) equal to the sum pf the present values of yearly activities (B):

$$PV(A) = SUM(PV(B)$$

$$A/(1+r)^n = B/(1+r)^1 + B/(1+r)^2 + \ldots + B/(1+r)^n$$

$$A/(1+r)^n = B*(1/(1+r)^1 + 1/(1+r)^2 + \ldots + 1/(1+r)^n)$$

$$A/(1+r)^n = B*((1+r)^(n-1) + (1+r)^(n-2) + \ldots + (1+r)^(n-n))$$

$$B = A/((1+r)^(n-1) + (1+r)^(n-2) + \ldots + (1+r)^(n-n))$$

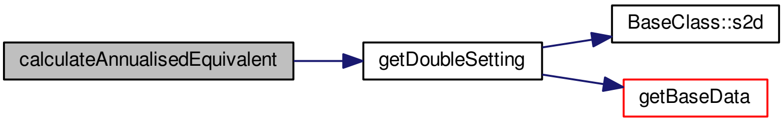
1. Changed for the equivalent but simpler eai = rev(t)∗i / ((1+i)^t-1)

Definition at line 1817 of file ModelData.cpp.

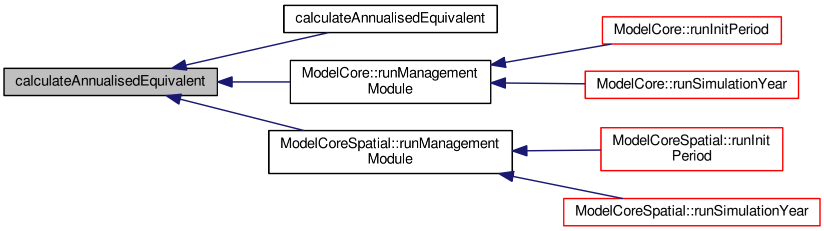
Referenced by calculateAnnualisedEquivalent(), ModelCore::runManagementModule(), and ModelCoreSpatial::runManagementModule().

```
01817                                                                    {
01818   // modified and tested 20120912. Before it was running this formula instead:
01819   // av.y.rev = rev(n)/ ( (1+ir)^1+(1+ir)^2+(1+ir)^3+...+(1+ir)^n )
01820   // the difference is that in this way the annual equivalent that is calulated doesn't need to be further
      discounted for yearly activites (e.g. agric)
01821
01822   //loop(fy$(ord(fy)=1),
01823   //  df(fy)= (1+ir)**(ord(fy));
01824   //);
01825   //loop(fy$(ord(fy)>1),
01826   //  df(fy)=df(fy-1)+(1+ir)**(ord(fy));
01827   //);
01828   if(years_h<0) return 0.;
01829   if(years_h==0) return amount_h;
01830   double ir = getDoubleSetting("ir");
01831   double eai = amount_h * ir / (pow(1.0+ir,years_h)-1.0);
01832
01833   return eai;
01834
01835     /*
01836   vector <double> df_by;
01837   for(int y=0;y<years_h;y++){
01838     double df;
01839     if(y==0){
01840       df = pow((1+ir),y);
01841     } else {
01842       df = df_by.at(y-1)+pow((1+ir),y);
01843     }
01844     if (y==years_h-1) {
01845          cout << eai << "    " << amount_h/df << endl;
01846      return amount_h/df; // big bug 20120904
01847     }
01848     df_by.push_back(df);
01849   }
01850   exit(1);
01851   return 0; // never reached, just to avoid compilation warnings
01852     */
01853 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


calculateAnnualisedEquivalent
calculateAnnualisedEquivalent
ModelCore::runManagement Module
ModelCoreSpatial::runManagement Module
ModelCore::runInitPeriod
ModelCore::runSimulationYear
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

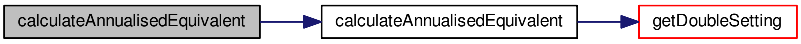
4.27.3.8 double calculateAnnualisedEquivalent (double *amount_h,* double *years_h*)

Transform the double to the highest integer and call calculateAnnualisedEquivalent(double amount_h, int years_h)

Definition at line 1856 of file ModelData.cpp.

```
01856                                                                           {
01857   //ceil(x) DNLP returns the smallest integer number greater than or equal to x
01858   //loop( (u,i,lambda,essence),
01859   //  cumTp(u,i,lambda,essence) =   ceil(cumTp(u,i,lambda,essence));
01860   //);
01861   int ceiledYear = ceil(years_h);
01862   return calculateAnnualisedEquivalent(amount_h, ceiledYear);
01863 }
```

Here is the call graph for this function:



4.27.3.9 vector< vector< int > > createCombinationsVector (const int & *nItems*)

Return a vector containing any possible combination of nItems items (including any possible subset). The returned vector has in each slot the items present in that specific combination.

ModelData::createCombinationsVector Return a vector containing any possible combination of nItems items (including all subsets).

For example with nItems = 3: 0: []; 1: [0]; 2: [1]; 3: [0,1]; 4: [2]; 5: [0,2]; 6: [1,2]; 7: [0,1,2]

Parameters

nItems	number of items to create p

Returns

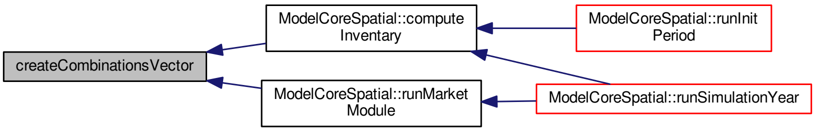
A vector with in each slot the items present in that specific combination subset.

Definition at line 1911 of file ModelData.cpp.

Referenced by ModelCoreSpatial::computeInventary(), and ModelCoreSpatial::runMarketModule().

```
01911                                                       {
01912   // Not confuse combination with permutation where order matter. Here it doesn't matter, as much as the
       algorithm is the same and returns
01913   // to as each position always the same subset
01914   vector < vector <int> > toReturn;
01915   int nCombs = pow(2,nItems);
01916   //int nCombs = nItems;
01917   for (uint i=0; i<nCombs; i++){
01918     vector<int> thisCombItems; //concernedPriProducts;
01919     for(uint j=0;j<nItems;j++){
01920       uint j2 = pow(2,j);
01921       if(i & j2){ // bit a bit operator, p217 C++ book
01922         thisCombItems.push_back(j);
01923       }
01924     }
01925     toReturn.push_back(thisCombItems);
01926   }
01927   return toReturn;
01928 }
```

Here is the caller graph for this function:


createCombinationsVector
ModelCoreSpatial::compute Inventary
ModelCoreSpatial::runMarket Module
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

4.27.3.10 void createRegions ( )

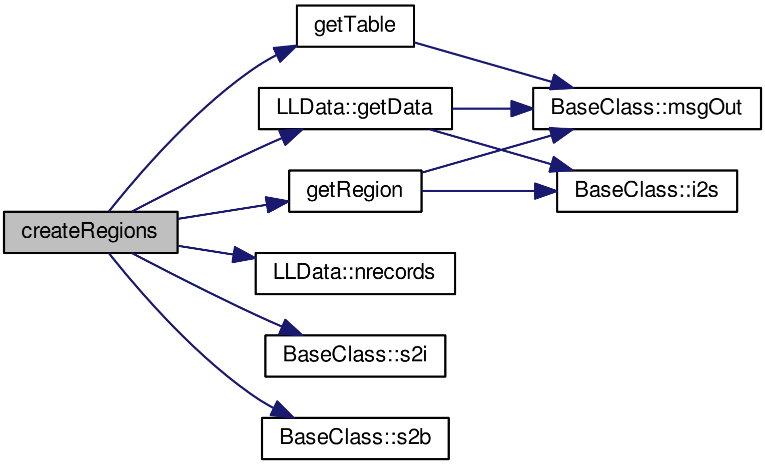
Definition at line 289 of file ModelData.cpp.

Referenced by Init::setInitLevel1().

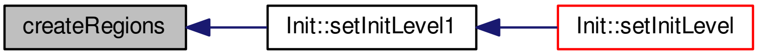
```
00289                         {
00290   // first create regions and assign basic data...
00291   LLData table = getTable("regions");
00292   for (int i=0; i< table.nrecords();i++){
00293     ModelRegion REGION(MTHREAD,
00294                 s2i(table.getData(i,"regId")),
00295                 table.getData(i,"regSName"),
00296                 table.getData(i,"regLName"),
00297                 s2i(table.getData(i,"regLevel")),
00298                 s2i(table.getData(i,"parRegId")),
00299                 s2b(table.getData(i,"isResidual")));
00300     regionsVector.push_back(REGION);
00301   }
00302   // Now let's assign the parent/children pointers..
```

```
  for (int i=0; i< regionsVector.size();i++){
    // let's assign the parent:
    regionsVector[i].setParent(this->getRegion(
regionsVector[i].getParRegId()));
    // let's assign the children:
    vector<ModelRegion*> kids;
    for (int y=0; y< regionsVector.size();y++){
      if(regionsVector[y].getParRegId() == regionsVector[i].getRegId() ){
        kids.push_back(&regionsVector[y]);
      }
    }
    regionsVector[i].setChildren(kids);
  }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



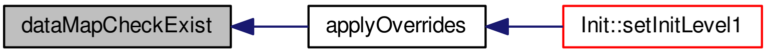
4.27.3.11 bool dataMapCheckExist (const DataMap & *map*, const string & *search_for*, const bool & *exactMatch* = `true`) const `[private]`

Definition at line 1668 of file ModelData.cpp.

Referenced by applyOverrides().

```
                                                                                                      {
  /*int dummyYear=MTHREAD->SCD->getYear();
  if(dataMapGetValue(map, search_for, dummyYear, exactMatch)==DATA_ERROR) {
    return false;
  } else {
    return true;
  }
  return false;
}*/
  bool found = false;
  DataMap::const_iterator i;
  if(!exactMatch){
    i = map.lower_bound(search_for);
    for(;i != map.end();i++){
      const string& key = i->first;
      if (key.compare(0, search_for.size(), search_for) == 0) {// Really a prefix?
          return true;
      } else {
        return false;
      }
    }
  } else {
    i = map.find(search_for);
    if (i!=map.end()){
      return true;
    }
  }
  return false;
}
```

Here is the caller graph for this function:



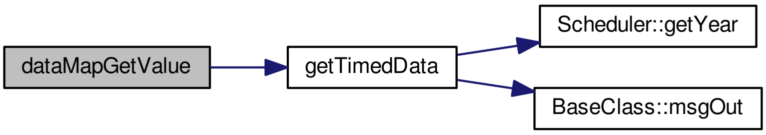
4.27.3.12 double dataMapGetValue (const DataMap & *map*, const string & *search_for*, const int & *year_h*, const bool & *exactMatch* = `true`) `[private]`

Definition at line 1700 of file ModelData.cpp.

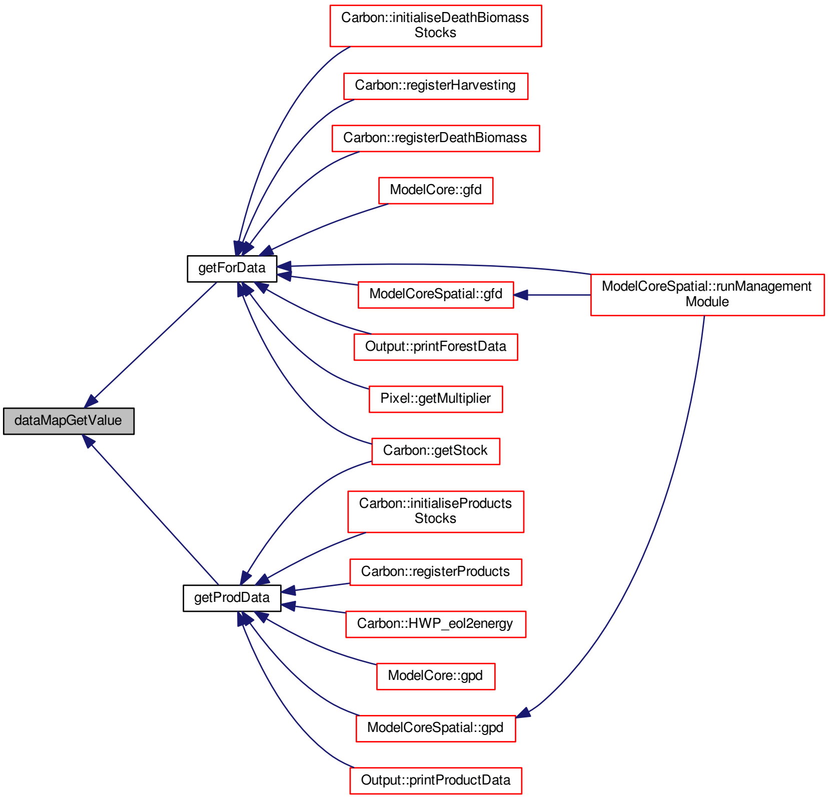
Referenced by getForData(), and getProdData().

```

        {
  double toReturn = 0;
  tempBool = false;
  DataMap::const_iterator i;
  if(!exactMatch){
    i = map.lower_bound(search_for);
    for(;i != map.end();i++){
      const string& key = i->first;
      if (key.compare(0, search_for.size(), search_for) == 0) {// Really a prefix?
          tempBool = true;
          toReturn += getTimedData( i->second, year_h );
      } else {
        break;
      }
    }
  } else {
    i = map.find(search_for);
    if (i!=map.end()){
      tempBool = true;
      return  getTimedData( i->second, year_h );
    }
  }
  return toReturn;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.3.13 int dataMapSetValue (DataMap & *map*, const string & *search_for*, const double & *value_h*, const int & *year_h*, const bool & *exactMatch* = `true`) `[private]`

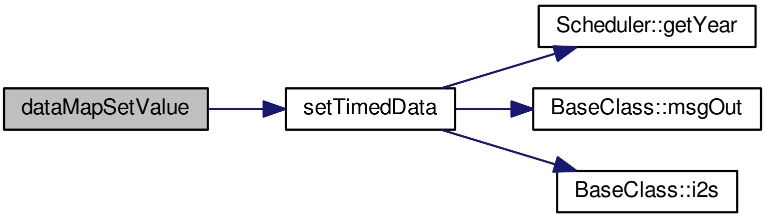
Definition at line 1728 of file ModelData.cpp.

Referenced by setForData(), and setProdData().

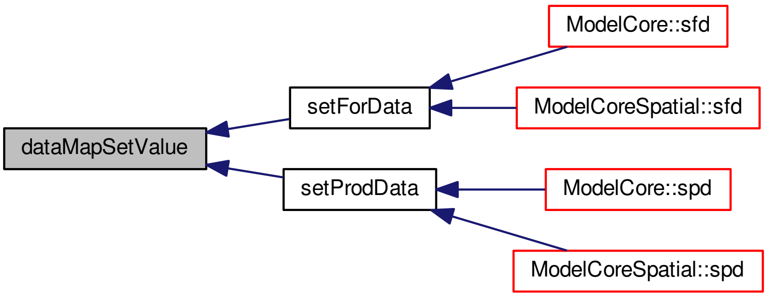
```
01728
                              {
01729   bool found = false;
01730   DataMap::iterator i;
```

```
  if(!exactMatch){
    i = map.lower_bound(search_for);
    for(;i != map.end();i++){
      const string& key = i->first;
      if (key.compare(0, search_for.size(), search_for) == 0) {// Really a prefix?
          found = true;
          setTimedData(value_h, i->second, year_h);
      } else {
        break;
      }
    }
  } else {
    i = map.find(search_for);
    if (i!=map.end()){
      found = true;
      setTimedData(value_h, i->second, year_h, errorLevel);
    }
  }
  // removed 20120903 as the insertion of new values must be explicitly done, not in all cases we want a new insertion
  /*if(!found){
    vector < double> newValues;
    setTimedData(value_h, newValues, year_h, MSG_NO_MSG); // don't warning if we are making a multi-year value vector, as it is a new one
    map.insert(DataPair (search_for,newValues));
  }*/
  return found;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



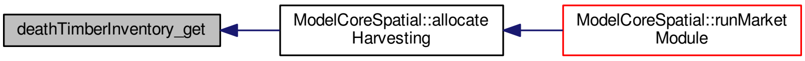
4.27.3.14 double deathTimberInventory_get (const iisskey & *thekey*) [inline]

Definition at line 190 of file ModelData.h.

Referenced by ModelCoreSpatial::allocateHarvesting().

```
00190 {return findMap(deathTimberInventory, thekey);}
```

Here is the caller graph for this function:


deathTimberInventory_get
ModelCoreSpatial::allocate Harvesting
ModelCoreSpatial::runMarket Module

4.27.3.15 void deathTimberInventory_incr (const iisskey & *thekey,* double *value_h*) [inline]

Definition at line 189 of file ModelData.h.

```
00189 {incrMapValue(deathTimberInventory,thekey, value_h);}
```

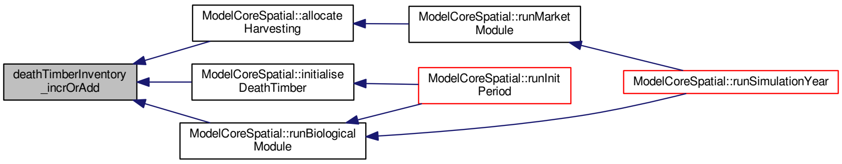
4.27.3.16 void deathTimberInventory_incrOrAdd (const iisskey & *thekey,* double *value_h*) [inline]

Definition at line 188 of file ModelData.h.

Referenced by ModelCoreSpatial::allocateHarvesting(), ModelCoreSpatial::initialiseDeathTimber(), and ModelCoreSpatial::runBiologicalModule().

```
00188 {incrOrAddMapValue(deathTimberInventory,thekey, value_h);}
```

Here is the caller graph for this function:


deathTimberInventory _incrOrAdd
ModelCoreSpatial::allocate Harvesting
ModelCoreSpatial::initialise DeathTimber
ModelCoreSpatial::runBiological Module
ModelCoreSpatial::runMarket Module
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

4.27.3.17 bool delDir (QString *dirname*)

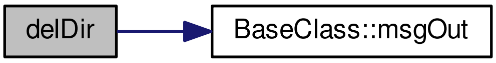
Recursivelly delete a directory.

Definition at line 1625 of file ModelData.cpp.

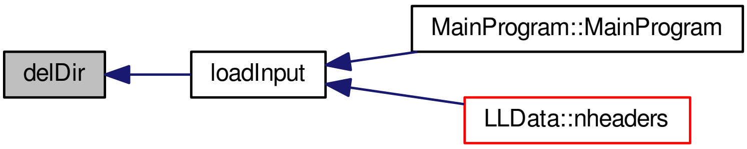
Referenced by loadInput().

```
01625                                  {
01626         bool deleted = false;
01627         QDir dir(dirname);
01628   //msgOut(MSG_DEBUG, dir.absolutePath().toStdString());
01629   dir.setFilter(QDir::Dirs | QDir::Files | QDir::NoDotAndDotDot | QDir::NoSymLinks);
01630   QFileInfoList list = dir.entryInfoList();
01631   deleted = dir.rmdir(dir.absolutePath());
01632   if (deleted) return true;
01633
01634   for (int i = 0; i < list.size(); ++i) {
01635     QFileInfo fileInfo = list.at(i);
01636     if (fileInfo.isFile()){
01637       //msgOut(MSG_DEBUG, "A file, gonna remove it: "+fileInfo.absoluteFilePath().toStdString());
01638       QFile targetFile(fileInfo.absoluteFilePath());
01639       bool fileDeleted = targetFile.remove();
01640       if (!fileDeleted){
01641         msgOut(MSG_CRITICAL_ERROR, "We have a problem: can't delete file "+fileInfo
      .absoluteFilePath().toStdString());
01642       }
01643     }
01644     else if (fileInfo.isDir()){
01645       //msgOut(MSG_DEBUG, "A directory, gonna go inside it: "+fileInfo.absoluteFilePath().toStdString());
01646       delDir(fileInfo.absoluteFilePath());
01647       dir.rmdir(fileInfo.absoluteFilePath());
01648     }
01649   }
01650
01651   deleted = dir.rmdir(dir.absolutePath());
01652   if (deleted) return true;
01653   return false;
01654 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.3.18 vector< int > getAllocableProductIdsFromDeathTimber (const int & *regId_h,* const string & *ft,* const string & *dc,* const int & *harvesting_year,* int *request_year* = DATA_NOW)

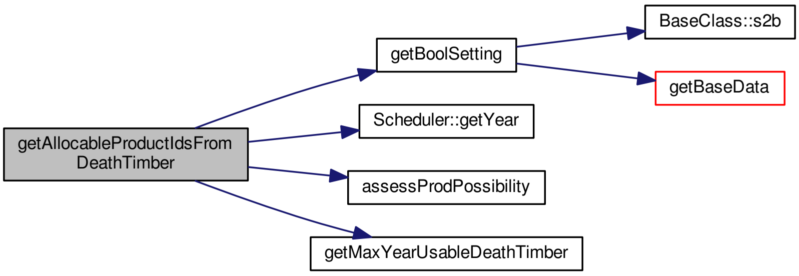
Returns the ids of the primary products that is possible to obtain using the timber recorded death in the specific year, ft, dc combination.

Definition at line 1961 of file ModelData.cpp.

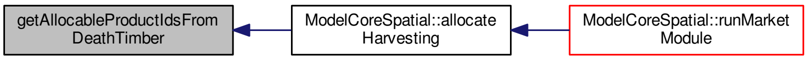
Referenced by ModelCoreSpatial::allocateHarvesting().

```
01961
                                                              {
01962   vector<int> allocableProductIds;
01963   if (!getBoolSetting("useDeathTimber")) return allocableProductIds;
01964   if (request_year == DATA_NOW) request_year = MTHREAD->SCD->
      getYear();
01965   for(uint p=0;p<priProducts.size();p++){
01966     string primProd = priProducts[p];
01967     if(assessProdPossibility(primProd,ft, dc)){
01968       int maxYears = getMaxYearUsableDeathTimber(primProd, ft, dc);
01969       if (request_year-harvesting_year < maxYears){
01970         allocableProductIds.push_back(p);
01971       }
01972     }
01973   }
01974   return allocableProductIds;
01975 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


getAllocableProductIdsFrom DeathTimber
ModelCoreSpatial::allocate Harvesting
ModelCoreSpatial::runMarket Module

4.27.3.19 vector< **ModelRegion** * > getAllRegions (bool *excludeResidual* = `true`)

Definition at line 351 of file ModelData.cpp.

```
00351                                          {
00352   vector <ModelRegion*> toReturn;
00353   for(uint i=0;i<regionsVector.size();i++){
00354     if( (!excludeResidual) || (!regionsVector[i].getIsResidual())){
00355       toReturn.push_back(&regionsVector[i]);
00356     }
00357   }
00358   return toReturn;
00359 }
```

4.27.3.20 double getAvailableAliveTimber (const vector< string > & *primProd_h*, int *regId_h*)

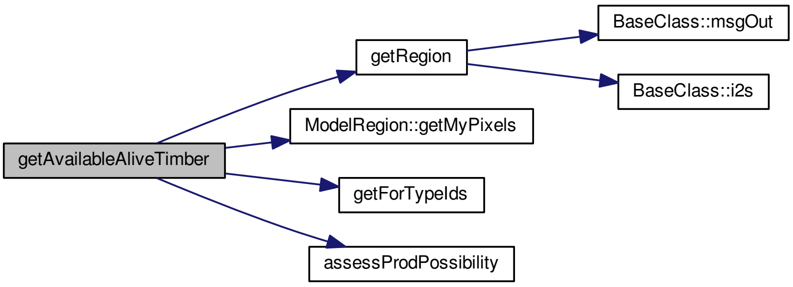
Returns the timber available for a given set of primary products as stored in the px->vol_l vector.

Definition at line 1980 of file ModelData.cpp.

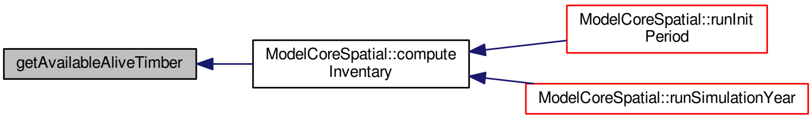
Referenced by ModelCoreSpatial::computeInventary().

```
01980                                                                                {
01981   double toReturn = 0.0;
01982   ModelRegion* REG = MTHREAD->MD->getRegion(regId_h);
01983   vector <Pixel*> regPx = REG->getMyPixels();
01984   vector <string> forTypesIds = getForTypeIds();
01985   for (uint i=0;i<forTypesIds.size();i++){
01986     string ft = forTypesIds[i];
01987     for(uint u=0;u<diamClasses.size();u++){
01988       string dc = diamClasses[u];
01989       bool possible = false;
01990       for (int p=0; p<primProd_h.size();p++){
01991           string primProd = primProd_h[p];
01992           if(assessProdPossibility(primProd,ft, dc)){
01993               possible = true;
01994           }
01995       }
01996       if(possible){
01997         for (uint p=0;p<regPx.size();p++){
01998           Pixel* px = regPx[p];
01999           toReturn += px->vol_l.at(i).at(u)*px->avalCoef;
02000         }
02001       }
02002     }
02003   }
02004   return toReturn;
02005 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


getAvailableAliveTimber
ModelCoreSpatial::compute Inventary
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

4.27.3.21 double getAvailableDeathTimber (const vector< string > & *primProd_h*, int *regID_h*, int *year_h*)

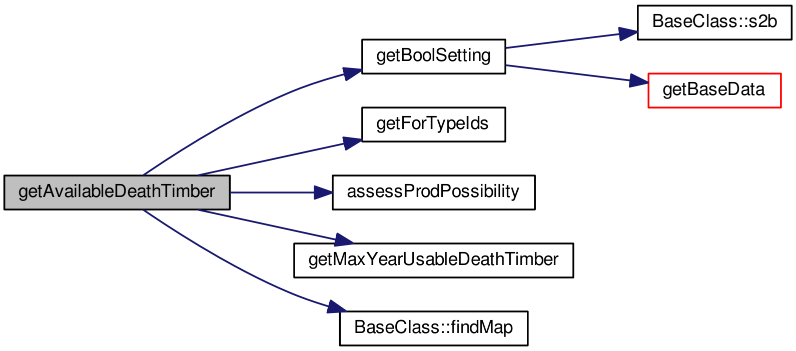
Returns the timber available for a given set of primary products as stored in the deathTimberInventory map.

Definition at line 1932 of file ModelData.cpp.

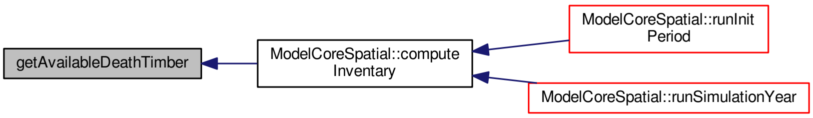
Referenced by ModelCoreSpatial::computeInventory().

```
01932                                                                                             {
01933   if (!getBoolSetting("useDeathTimber")) return 0;
01934   double toReturn = 0.0;
01935   vector <string> forTypesIds = getForTypeIds();
01936   for (uint i=0;i<forTypesIds.size();i++){
01937     string ft = forTypesIds[i];
01938     for(uint u=0;u<diamClasses.size();u++){
01939       string dc = diamClasses[u];
01940       bool possible = false;
01941       int maxYears = 0;
01942       for (int p=0; p<primProd_h.size();p++){
01943           string primProd = primProd_h[p];
01944           if(assessProdPossibility(primProd,ft, dc)){
01945               possible = true;
01946               maxYears=max(maxYears,getMaxYearUsableDeathTimber(primProd, ft, dc
      ));
01947           }
01948       }
01949       if(possible){
01950         for(int y=year_h;y>year_h-maxYears;y--){
01951           iisskey key(y,regId_h,ft,dc);
01952           toReturn += findMap(deathTimberInventory,key,
      MSG_DEBUG,0.0);
01953         }
01954       }
01955     }
01956   }
01957   return toReturn;
01958 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


getAvailableDeathTimber
ModelCoreSpatial::compute Inventory
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

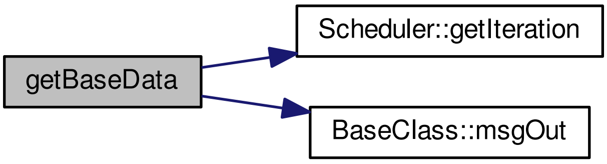
4.27.3.22 string getBaseData (const string & *name_h,* int *type_h,* int *position =* `0`) `[private]`

Definition at line 955 of file ModelData.cpp.

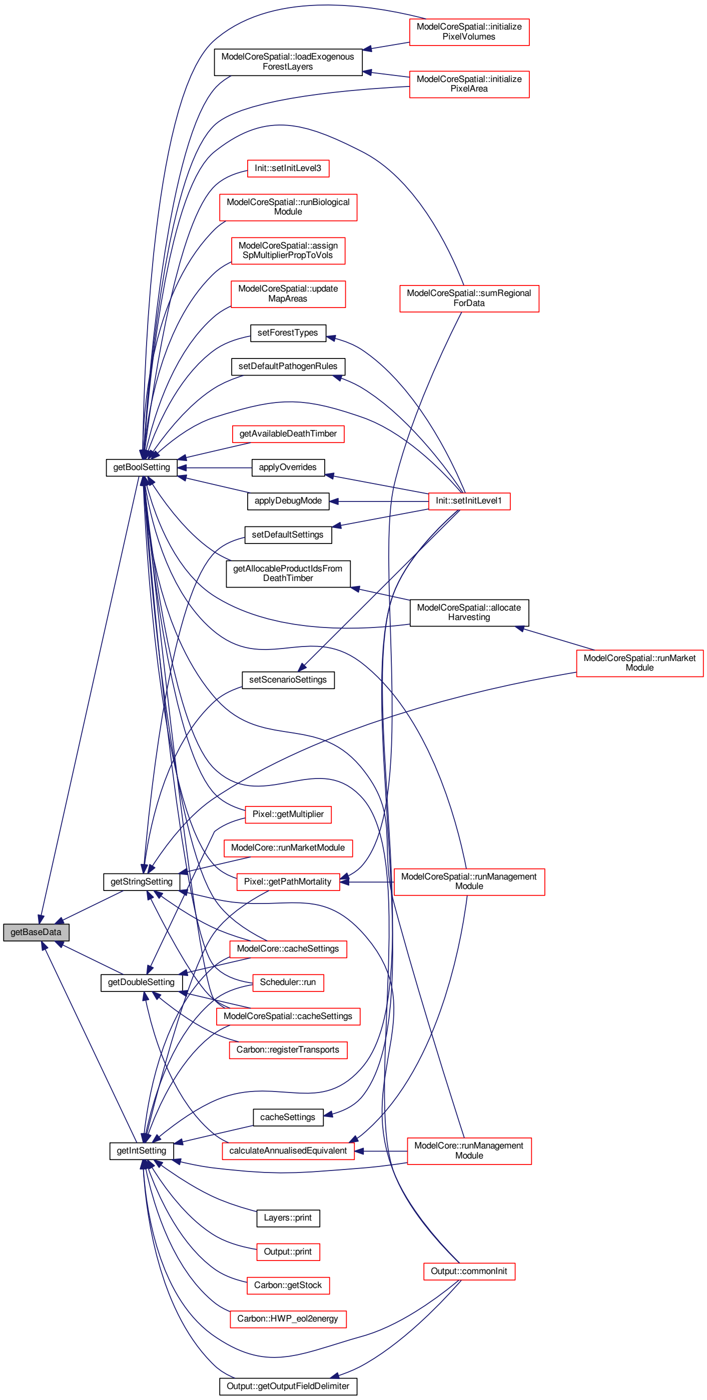
Referenced by getBoolSetting(), getDoubleSetting(), getIntSetting(), and getStringSetting().

```
00955                                                                          {
00956   // If the data is called with DATA_NOW we interpret the array of values as a temporal array and we return
       the value at the current time.
00957   if(position == DATA_NOW) {
00958     position = MTHREAD->SCD->getIteration();
00959   }
00960   for (uint i=0; i<programSettingsVector.size();i++){
00961     if (programSettingsVector.at(i).name == name_h){
00962       int type = programSettingsVector.at(i).type;
00963       if(type != type_h){msgOut(MSG_CRITICAL_ERROR, "mismatching type in calling
       getBaseData() for "+name_h);}
00964       if(programSettingsVector.at(i).values.size() > ((uint)position)) {
00965         return programSettingsVector.at(i).values.at(position);
00966       } else if (programSettingsVector.at(i).values.size() > 0 ){
00967         // returning the last available value...
00968         return programSettingsVector.at(i).values.at(
      programSettingsVector.at(i).values.size()-1 );
00969       }
00970       else {msgOut(MSG_CRITICAL_ERROR, "Error: "+name_h+" doesn't have any value,
       even on the first position(year)!"); }
00971     }
00972   }
00973   if(type_h==TYPE_BOOL){
00974     msgOut(MSG_DEBUG, "Possible error calling getBaseData(TYPE_BOOL) for "+ name_h +". No
       setting option or macro data found with this name. Returning false.");
00975     return "0";
00976   } else {
00977     msgOut(MSG_CRITICAL_ERROR, "Error calling getBaseData() for "+ name_h +". No
       setting option or macro data found with this name.");
00978   }
00979   return "";
00980 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


ModelCoreSpatial::initializePixelVolumes
ModelCoreSpatial::loadExogenousForestLayers
ModelCoreSpatial::initializePixelArea
Init::setInitLevel3
ModelCoreSpatial::runBiologicalModule
ModelCoreSpatial::assignSpMultiplierPropToVols
ModelCoreSpatial::updateMapAreas
ModelCoreSpatial::sumRegionalForData
setForestTypes
setDefaultPathogenRules
getAvailableDeathTimber
getBoolSetting
applyOverrides
applyDebugMode
Init::setInitLevel1
setDefaultSettings
getAllocableProductIdsFromDeathTimber
ModelCoreSpatial::allocateHarvesting
ModelCoreSpatial::runMarketModule
setScenarioSettings
Pixel::getMultiplier
ModelCore::runMarketModule
getStringSetting
Pixel::getPathMortality
ModelCoreSpatial::runManagementModule
getBaseData
ModelCore::cacheSettings
getDoubleSetting
Scheduler::run
ModelCoreSpatial::cacheSettings
Carbon::registerTransports
cacheSettings
getIntSetting
calculateAnnualisedEquivalent
ModelCore::runManagementModule
Layers::print
Output::print
Output::commonInit
Carbon::getStock
Carbon::HWP_eol2energy
Output::getOutputFieldDelimiter

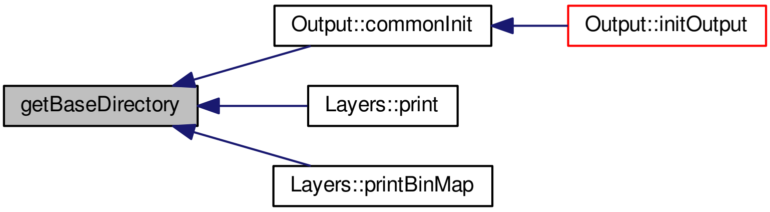
4.27.3.23 string getBaseDirectory ( ) const [inline]

Definition at line 116 of file ModelData.h.

Referenced by Output::commonInit(), Layers::print(), and Layers::printBinMap().

```
00116 {return baseDirectory;}
```

Here is the caller graph for this function:



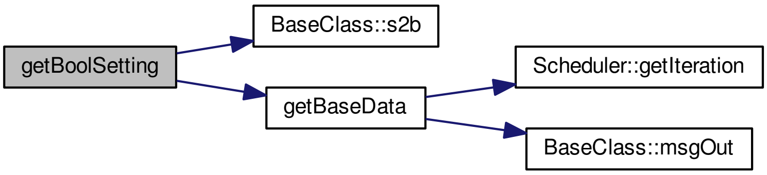
4.27.3.24 bool getBoolSetting (const string & *name_h*, int *position* = 0) const

Definition at line 1010 of file ModelData.cpp.

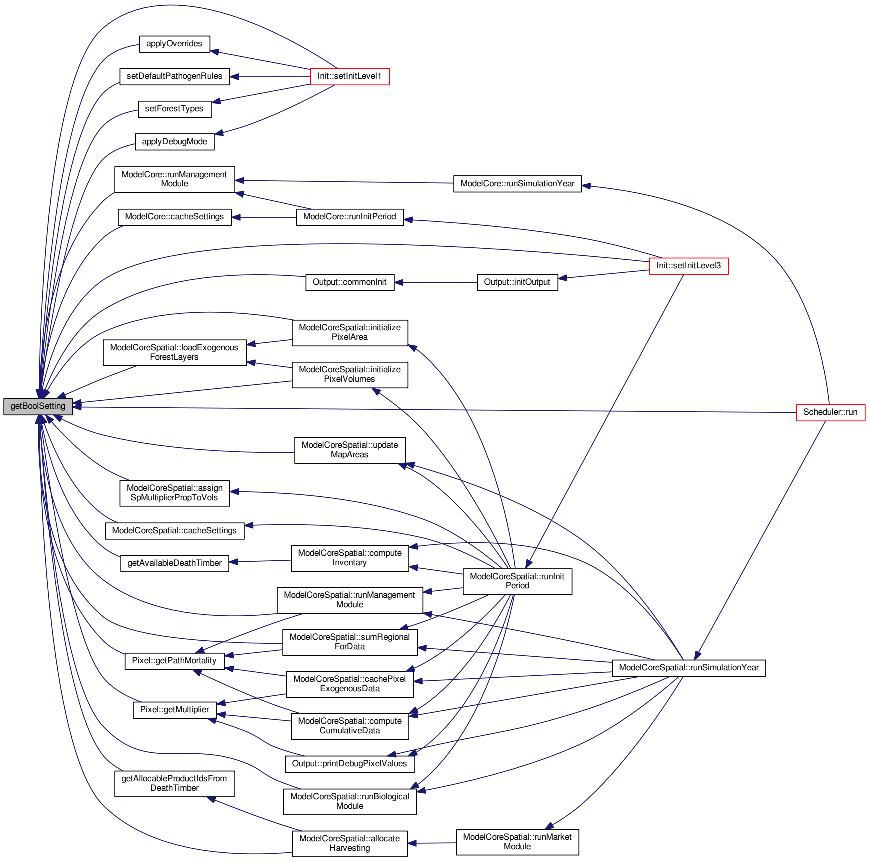
Referenced by ModelCoreSpatial::allocateHarvesting(), applyDebugMode(), applyOverrides(), ModelCoreSpatial::assignSpMultiplierPropToVols(), ModelCore::cacheSettings(), ModelCoreSpatial::cacheSettings(), Output::commonInit(), getAllocableProductIdsFromDeathTimber(), getAvailableDeathTimber(), Pixel::getMultiplier(), Pixel::getPathMortality(), ModelCoreSpatial::initializePixelArea(), ModelCoreSpatial::initializePixelVolumes(), ModelCoreSpatial::loadExogenousForestLayers(), Scheduler::run(), ModelCoreSpatial::runBiologicalModule(), ModelCore::runManagementModule(), ModelCoreSpatial::runManagementModule(), setDefaultPathogenRules(), setForestTypes(), Init::setInitLevel1(), Init::setInitLevel3(), ModelCoreSpatial::sumRegionalForData(), and ModelCoreSpatial::updateMapAreas().

```
01010                                                             {
01011   return s2b( MTHREAD->MD->getBaseData(name_h,TYPE_BOOL,position) );
01012 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

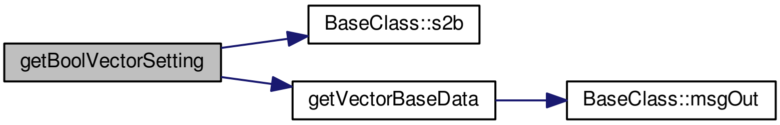

applyOverrides
setDefaultPathogenRules
Init::setInitLevel1
setForestTypes
applyDebugMode
ModelCore::runManagement Module
ModelCore::runSimulationYear
ModelCore::cacheSettings
ModelCore::runInitPeriod
Init::setInitLevel3
Output::commonInit
Output::initOutput
ModelCoreSpatial::initialize PixelArea
ModelCoreSpatial::loadExogenous ForestLayers
ModelCoreSpatial::initialize PixelVolumes
getBoolSetting
Scheduler::run
ModelCoreSpatial::update MapAreas
ModelCoreSpatial::assign SpMultiplierPropToVols
ModelCoreSpatial::cacheSettings
getAvailableDeathTimber
ModelCoreSpatial::compute Inventary
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::sumRegional ForData
Pixel::getPathMortality
ModelCoreSpatial::runSimulationYear
ModelCoreSpatial::cachePixel ExogenousData
Pixel::getMultiplier
ModelCoreSpatial::compute CumulativeData
Output::printDebugPixelValues
getAllocableProductIdsFrom DeathTimber
ModelCoreSpatial::runBiological Module
ModelCoreSpatial::allocate Harvesting
ModelCoreSpatial::runMarket Module

4.27.3.25 vector< bool > getBoolVectorSetting (const string & *name_h*) const

Definition at line 1026 of file ModelData.cpp.

```
01026                                                               {
01027   return s2b(MTHREAD->MD->getVectorBaseData(name_h,
     TYPE_BOOL));
01028 }
```

Here is the call graph for this function:



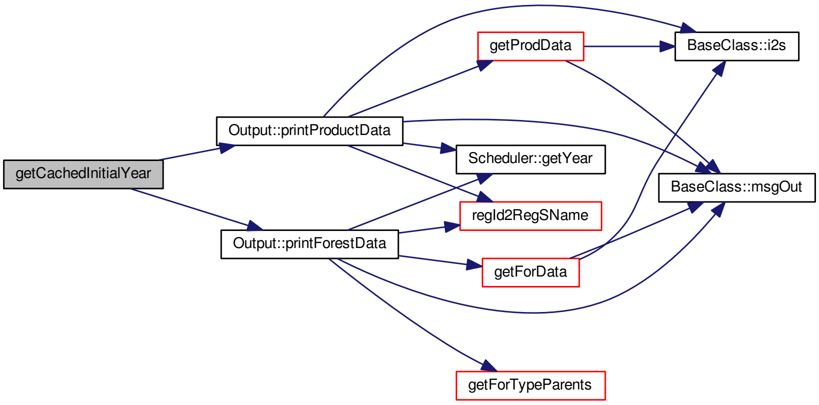
4.27.3.26 int getCachedInitialYear ( ) [inline]

Definition at line 180 of file ModelData.h.

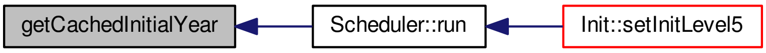
Referenced by Scheduler::run().

```
00180 {return cached_initialYear;}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.3.27 map<iisskey, double >* getDeathTimberInventory ( ) [inline]

Definition at line 191 of file ModelData.h.

```
00191 {return &deathTimberInventory;};
```

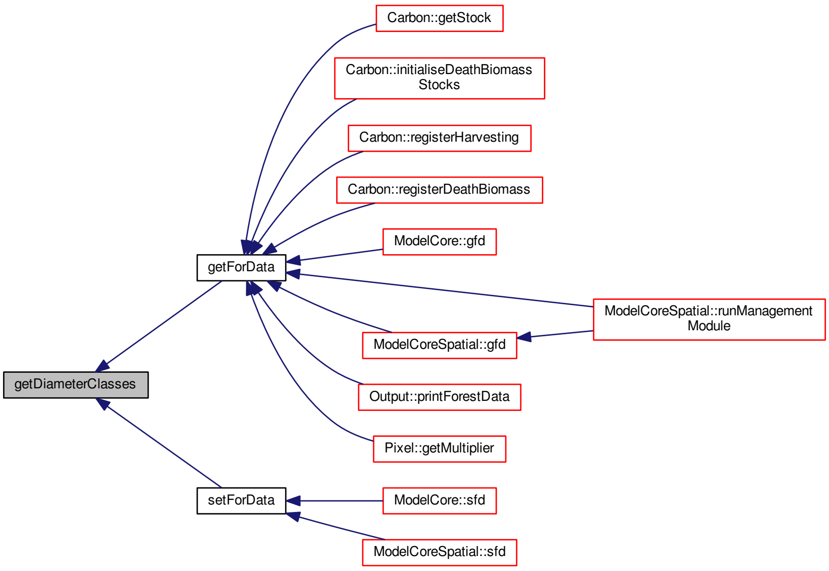
4.27.3.28 vector< string > getDiameterClasses (bool *productionOnly* = false)

Definition at line 1083 of file ModelData.cpp.

Referenced by getForData(), and setForData().

```
01083                                               {
01084   int i;
01085   if(productionOnly){
01086     i=1;
01087   } else {
01088     i=0;
01089   }
01090   vector <string> toReturn;
01091   for (i;i<diamClasses.size();i++){
01092     toReturn.push_back(diamClasses[i]);
01093   }
01094   return toReturn;
01095 }
```

Here is the caller graph for this function:



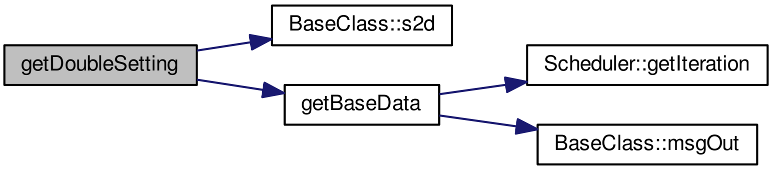
4.27.3.29 double getDoubleSetting (const string & *name_h*, int *position* = 0) const

Definition at line 1002 of file ModelData.cpp.

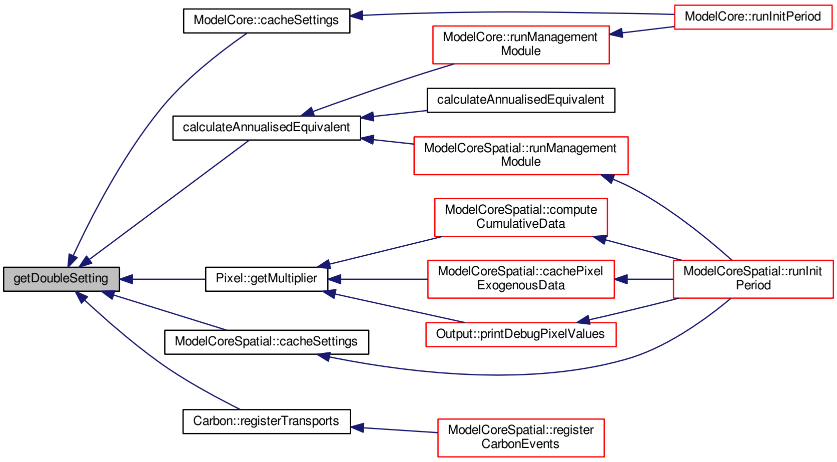
Referenced by ModelCore::cacheSettings(), ModelCoreSpatial::cacheSettings(), calculateAnnualisedEquivalent(), Pixel::getMultiplier(), and Carbon::registerTransports().

```
01002                                                             {
01003   return s2d( MTHREAD->MD->getBaseData(name_h,TYPE_DOUBLE,position) );
01004 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

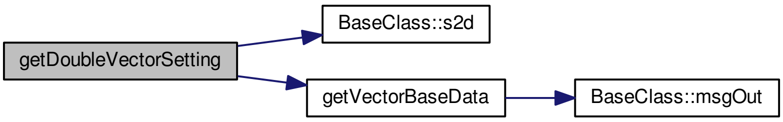

ModelCore::cacheSettings
ModelCore::runInitPeriod
ModelCore::runManagement Module
calculateAnnualisedEquivalent
calculateAnnualisedEquivalent
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::compute CumulativeData
getDoubleSetting
Pixel::getMultiplier
ModelCoreSpatial::cachePixel ExogenousData
ModelCoreSpatial::runInit Period
Output::printDebugPixelValues
ModelCoreSpatial::cacheSettings
Carbon::registerTransports
ModelCoreSpatial::register CarbonEvents

4.27.3.30 vector< double > getDoubleVectorSetting (const string & *name_h*) const

Definition at line 1018 of file ModelData.cpp.

```
01018                                                     {
01019   return s2d(MTHREAD->MD->getVectorBaseData(name_h,
      TYPE_DOUBLE));
01020 }
```

Here is the call graph for this function:



4.27.3.31 std::string getFilenameByType (std::string *type_h*)

Definition at line 1067 of file ModelData.cpp.

```
01067                                             {
01068   std::string directory;
01069   std::string filename;
01070   std::string filename_complete;
01071   for (uint i=0; i<iFilesVector.size(); i++){
01072     if (iFilesVector.at(i).type == type_h){
01073       directory=iFilesVector.at(i).directory;
01074       filename=iFilesVector.at(i).name;
01075       break;
01076     }
01077   }
01078   filename_complete = baseDirectory+directory+filename;
01079   return filename_complete;
01080 }
```

4.27.3.32 int getFilenamesByDir (const string & *dir*, vector< string > & *files*, const string & *filter* = "")

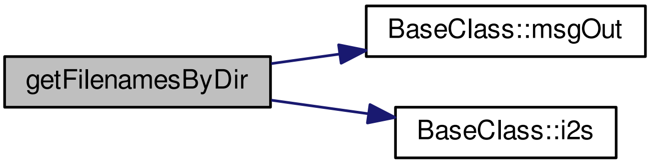
Return a list of files in a directory.

Get a list of files in a directory

Definition at line 1867 of file ModelData.cpp.

```
01867                                                                                                {
01868   DIR *dp;
01869   struct dirent *dirp;
01870   if((dp  = opendir(dir.c_str())) == NULL) {
01871     msgOut(MSG_ERROR, "Error " + i2s(errno) + " opening the " + dir + " directory.");
01872     //cout << "Error(" << errno << ") opening " << dir << endl;
01873     return errno;
01874   }
01875   while ((dirp = readdir(dp)) != NULL) {
01876     string filename = dirp->d_name;
01877     if(
01878       (filter != "" && filename.substr(filename.find_last_of(".")) == filter) // there is a filter and the
       last bit of the filename match the filter
01879       || (filter == "" && filename.substr(filename.find_last_of(".") + 1) != "") // there isn't any filter
       but we don't want stuff like ".." or "."
01880     ) {
01881       files.push_back(string(dirp->d_name));
01882     }
01883   }
01884   closedir(dp);
01885   return 0;
01886 }
```

Here is the call graph for this function:



4.27.3.33 const double getForData (const string & *type_h,* const int & *regId_h,* const string & *forType_h,* const string & *freeDim_h,* const int & *year* = DATA_NOW)

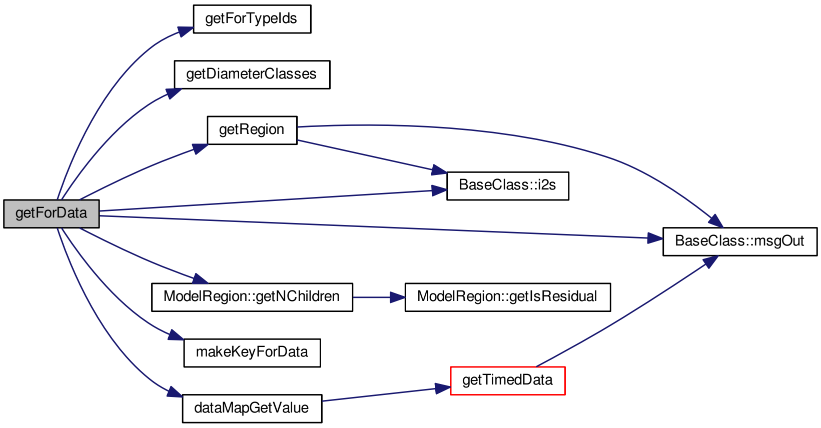
Basic function to retrieve forest-related data. It addmits the following "filters": Name of the specific parameter requested Look for a level1 or level2 region If specified, look exactly for the specified forest type, otherwise accept the keyword FT_ALL for summing all of them Normally used for diameter class, but occasionally used for other uses (changed 20140514). It accepts three keywords, for summing up all diameters, production-ready diameters or sub-production ones, namelly DIAM_ALL, DIAM_PROD, DIAM_FIRST.\ If a diameter-independed variable is required, put it in the data with an empty diameter class and retrieve it here using DIAM_ALL. Unless specified, get the value of the current year. If array is smaller (e.g. because it is time-independent), get the last value.

Definition at line 1172 of file ModelData.cpp.

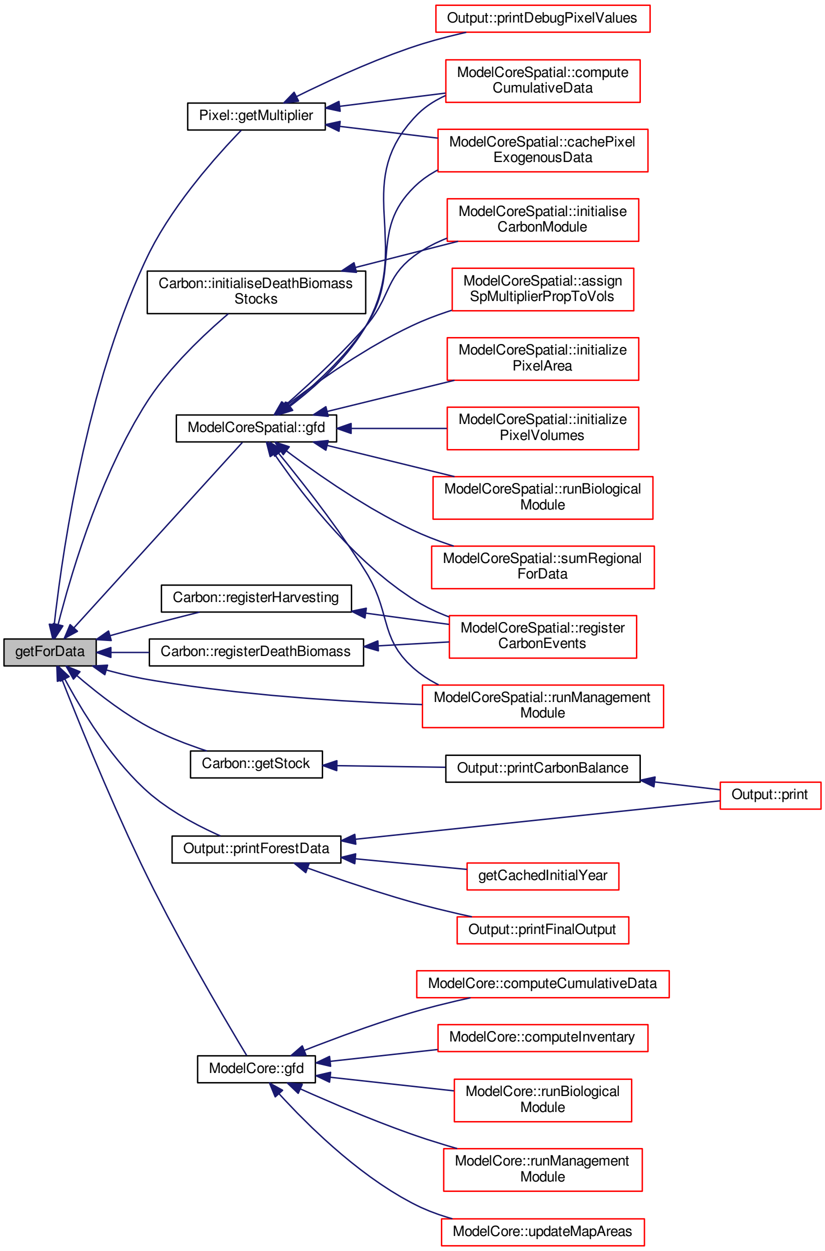
Referenced by Pixel::getMultiplier(), Carbon::getStock(), ModelCore::gfd(), ModelCoreSpatial::gfd(), Carbon::initialiseDeathBiomassStocks(), Output::printForestData(), Carbon::registerDeathBiomass(), Carbon::registerHarvesting(), and ModelCoreSpatial::runManagementModule().

```
01172
                           {
01173   vector<int> regIds;
01174   vector <string> dClasses;
01175   vector <string> fTypes;
01176   string key;
01177   DataMap::const_iterator p;
01178   bool found = false;
01179   double value = 0;
01180
01181   // creating the arrays to look up if keywords were specified..
01182   if (forType_h == FT_ALL){ // || forType_h == ""){
01183     fTypes = getForTypeIds();
01184     fTypes.push_back("");
01185   } else {
01186     fTypes.push_back(forType_h);
01187   }
01188   if(freeDim_h == DIAM_ALL){ // || freeDim_h == ""){
01189     dClasses = diamClasses;
01190     dClasses.push_back("");
01191   } else if (freeDim_h == DIAM_PROD){
01192     dClasses = getDiameterClasses(true);
01193   } else if (freeDim_h == DIAM_FIRST){
01194     dClasses.push_back(diamClasses.at(0));
01195   } else  {
01196     dClasses.push_back(freeDim_h);
01197   }
01198   // Make sure to set the new value to all l2 regions if requested for a reg1 level
01199   if(getRegion(regId_h)->getRegLevel()==2){
01200     regIds.push_back(regId_h);
01201   } else if (getRegion(regId_h)->getRegLevel()==1)  {
01202     for(uint i=0;i<getRegion(regId_h)->getNChildren();i++){
01203       regIds.push_back(getRegion(regId_h)->getChildren()[i]->getRegId());
01204     }
01205   } else {
01206     msgOut(MSG_CRITICAL_ERROR, "Error in getProdData(). Setting a value for the
       whole World is not supported.");
01207   }
01208   int regIdsS = regIds.size();
01209
01210   // getting the actual data...
01211   for(uint r=0;r< regIds.size();r++){
01212     for(uint i=0;i<dClasses.size();i++){
01213       for (uint y=0;y<fTypes.size();y++){
01214         key = makeKeyForData(type_h,i2s(regIds[r]),fTypes[y],dClasses[i]);
01215         value += dataMapGetValue(forDataMap,key,year,true);
01216         if(tempBool) found = true;
01217       }
01218     }
01219   }
01220
01221   if(!found){
01222         msgOut(errorLevel, "Error in getForData(): no combination found for "+type_h+", "+
      i2s(regId_h)+", "+forType_h+", "+i2s(year)+", "+freeDim_h+". Returning 0, but double check that this
       is ok for your model.");
01223   }
01224   return value;
01225 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


Output::printDebugPixelValues
ModelCoreSpatial::compute CumulativeData
Pixel::getMultiplier
ModelCoreSpatial::cachePixel ExogenousData
ModelCoreSpatial::initialise CarbonModule
Carbon::initialiseDeathBiomass Stocks
ModelCoreSpatial::assign SpMultiplierPropToVols
ModelCoreSpatial::initialize PixelArea
ModelCoreSpatial::gfd
ModelCoreSpatial::initialize PixelVolumes
ModelCoreSpatial::runBiological Module
ModelCoreSpatial::sumRegional ForData
Carbon::registerHarvesting
ModelCoreSpatial::register CarbonEvents
getForData
Carbon::registerDeathBiomass
ModelCoreSpatial::runManagement Module
Carbon::getStock
Output::printCarbonBalance
Output::print
Output::printForestData
getCachedInitialYear
Output::printFinalOutput
ModelCore::computeCumulativeData
ModelCore::computeInventary
ModelCore::gfd
ModelCore::runBiological Module
ModelCore::runManagement Module
ModelCore::updateMapAreas

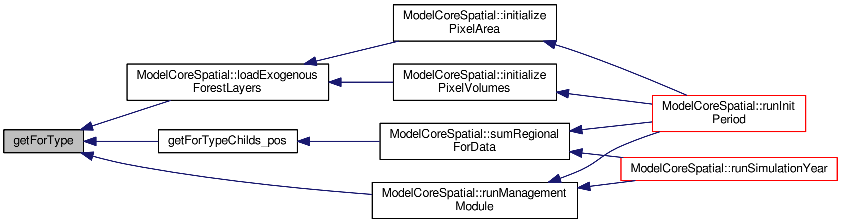
4.27.3.34 forType∗ getForType (int *position*) [inline]

Definition at line 126 of file ModelData.h.

Referenced by getForTypeChilds_pos(), ModelCoreSpatial::loadExogenousForestLayers(), and ModelCoreSpatial::runManagementModule().

```
00126 {return &forTypes[position];}
```

Here is the caller graph for this function:

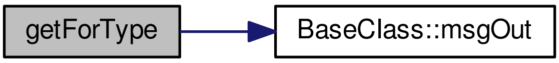

ModelCoreSpatial::initialize PixelArea
ModelCoreSpatial::loadExogenous ForestLayers
ModelCoreSpatial::initialize PixelVolumes
ModelCoreSpatial::runInit Period
getForType
getForTypeChilds_pos
ModelCoreSpatial::sumRegional ForData
ModelCoreSpatial::runSimulationYear
ModelCoreSpatial::runManagement Module

4.27.3.35 forType ∗ getForType (string & *forTypeId_h*)

Definition at line 71 of file ModelData.cpp.

```
00071                                         {
00072   for(int i=0;i<forTypes.size();i++){
00073     if(forTypes[i].forTypeId==forTypeId_h) return &forTypes[i];
00074   }
00075   msgOut(MSG_CRITICAL_ERROR,"forTypeId "+forTypeId_h+" not found. Aborting.");
00076 }
```

Here is the call graph for this function:



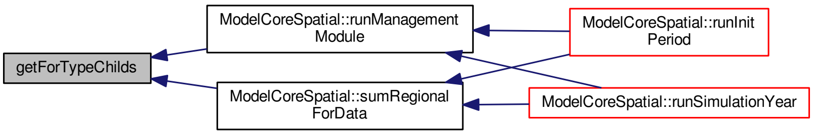
4.27.3.36 vector< string > getForTypeChilds (const string & *forTypeId_h*)

Definition at line 96 of file ModelData.cpp.

Referenced by ModelCoreSpatial::runManagementModule(), and ModelCoreSpatial::sumRegionalForData().

```
00096                                                     {
00097   vector<string> childs;
00098   for(int i=0;i<forTypes.size();i++){
00099       if(forTypes[i].ereditatedFrom==forTypeId_h) {
00100           childs.push_back(forTypes[i].forTypeId);
00101       }
00102   }
00103   return childs;
00104 }
```

Here is the caller graph for this function:


ModelCoreSpatial::runManagement Module
ModelCoreSpatial::runInit Period
getForTypeChilds
ModelCoreSpatial::sumRegional ForData
ModelCoreSpatial::runSimulationYear

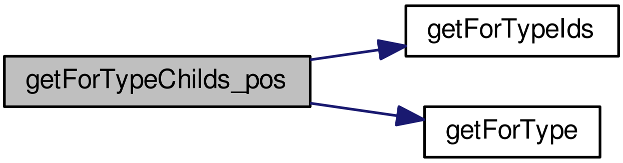
4.27.3.37 vector< int > getForTypeChilds_pos (const string & *forTypeId_h,* bool *all* = `false`)

Definition at line 107 of file ModelData.cpp.

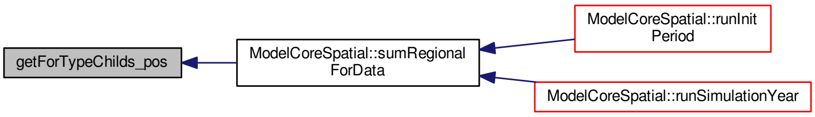
Referenced by ModelCoreSpatial::sumRegionalForData().

```
00107                                                                 {
00108   vector <int> childs;
00109   vector <string> fTIds = getForTypeIds(all);
00110   for(int i=0;i<fTIds.size();i++){
00111       forType* ft = getForType(fTIds[i]);
00112       if(ft->ereditatedFrom==forTypeId_h) {
00113           childs.push_back(i);
00114       }
00115   }
00116   return childs;
00117 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


getForTypeChilds_pos
ModelCoreSpatial::sumRegional ForData
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

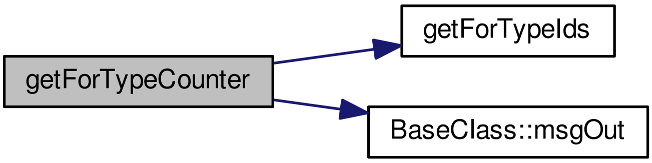
4.27.3.38 int getForTypeCounter (string & *forTypeId_h,* bool *all* = `false`)

By default it doesn't return forTypes used only as input.

Definition at line 79 of file ModelData.cpp.

```
00079                                                         {
00080   vector <string> fTIds = getForTypeIds(all);
00081   for(int i=0;i<fTIds.size();i++){
00082     if(fTIds[i]==forTypeId_h) return i;
00083   }
00084   msgOut(MSG_CRITICAL_ERROR,"forTypeId "+forTypeId_h+" not found in "+((string)
       __func__ )+". Aborting.");
00085 }
```

Here is the call graph for this function:



4.27.3.39 vector< string > getForTypeIds (bool *all* = `false`)

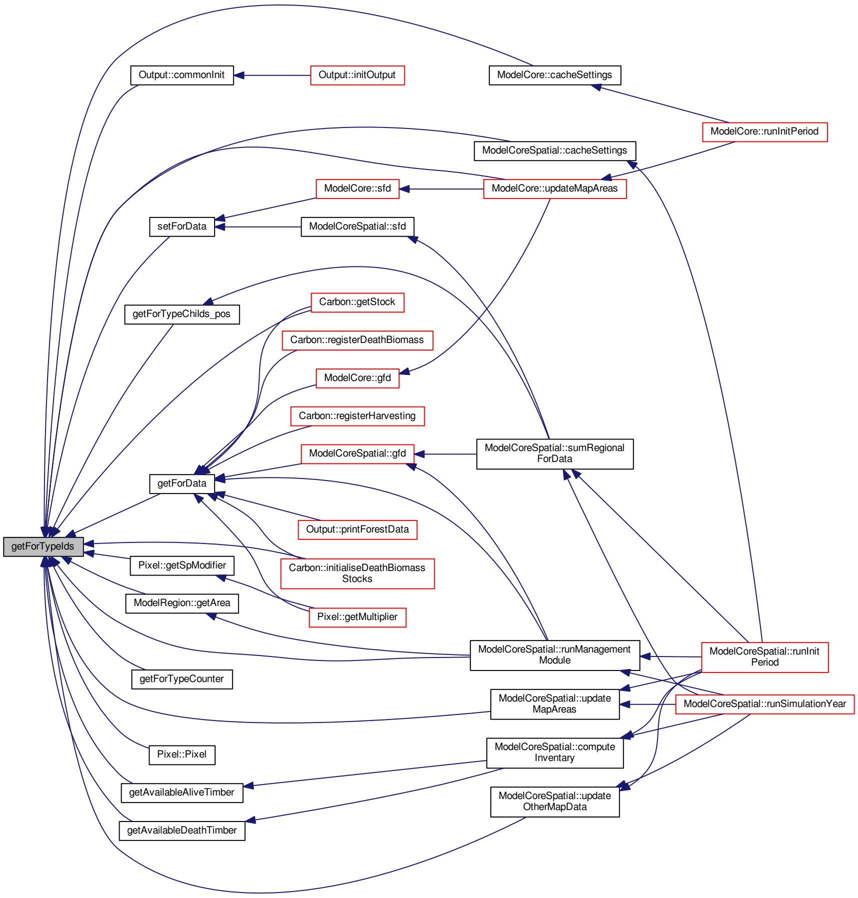
By default it doesn't return forTypes used only as input.

Definition at line 402 of file ModelData.cpp.

Referenced by ModelCore::cacheSettings(), ModelCoreSpatial::cacheSettings(), Output::commonInit(), ModelRegion::getArea(), getAvailableAliveTimber(), getAvailableDeathTimber(), getForData(), getForTypeChilds_pos(), getForTypeCounter(), Pixel::getSpModifier(), Carbon::getStock(), Carbon::initialiseDeathBiomassStocks(), Pixel::Pixel(), ModelCoreSpatial::runManagementModule(), setForData(), ModelCore::updateMapAreas(), ModelCoreSpatial::updateMapAreas(), and ModelCoreSpatial::updateOtherMapData().

```
                                        {
  vector <string> toReturn;
  for(uint i=0;i<forTypes.size();i++){
    if(forTypes[i].memType!=1 || all) {
      toReturn.push_back(forTypes[i].forTypeId);
    }
  }
  return toReturn;
}
```

Here is the caller graph for this function:


Output::commonInit
Output::initOutput
ModelCore::cacheSettings
ModelCore::runInitPeriod
ModelCoreSpatial::cacheSettings
ModelCore::sfd
ModelCore::updateMapAreas
setForData
ModelCoreSpatial::sfd
getForTypeChilds_pos
Carbon::getStock
Carbon::registerDeathBiomass
ModelCore::gfd
Carbon::registerHarvesting
ModelCoreSpatial::gfd
ModelCoreSpatial::sumRegional ForData
getForData
Output::printForestData
getForTypeIds
Pixel::getSpModifier
Carbon::initialiseDeathBiomass Stocks
ModelRegion::getArea
Pixel::getMultiplier
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::runInit Period
getForTypeCounter
ModelCoreSpatial::update MapAreas
ModelCoreSpatial::runSimulationYear
Pixel::Pixel
ModelCoreSpatial::compute Inventary
getAvailableAliveTimber
ModelCoreSpatial::update OtherMapData
getAvailableDeathTimber

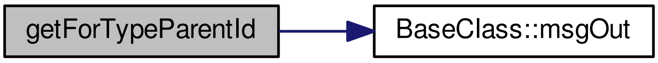
4.27.3.40 string getForTypeParentId (const string & *forTypeId_h*)

Definition at line 88 of file ModelData.cpp.

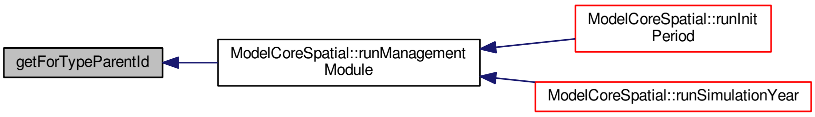
Referenced by ModelCoreSpatial::runManagementModule().

```
00088                                                         {
00089     for(int i=0;i<forTypes.size();i++){
00090         if(forTypes[i].forTypeId==forTypeId_h) return forTypes[i].ereditatedFrom;
00091     }
00092     msgOut(MSG_CRITICAL_ERROR,"forTypeId "+forTypeId_h+" not found. Aborting.");
00093 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


getForTypeParentId
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

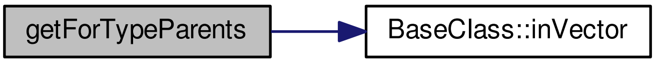
4.27.3.41 vector< string > getForTypeParents ( )

Definition at line 120 of file ModelData.cpp.

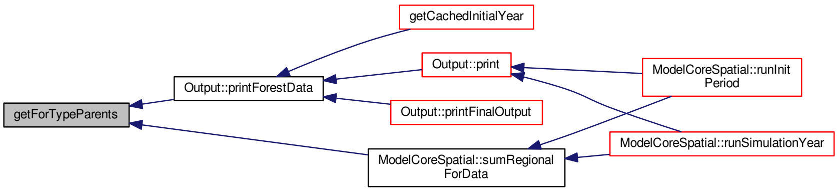
Referenced by Output::printForestData(), and ModelCoreSpatial::sumRegionalForData().

```
00120                          {
00121   vector<string> parents;
00122   for(int i=0;i<forTypes.size();i++){
00123     string parent = forTypes[i].ereditatedFrom;
00124     if(!inVector(parent,parents) && parent != ""){
00125       parents.push_back(parent);
00126     }
00127   }
00128   return parents;
00129 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


getCachedInitialYear
Output::print
ModelCoreSpatial::runInit Period
Output::printForestData
getForTypeParents
Output::printFinalOutput
ModelCoreSpatial::runSimulationYear
ModelCoreSpatial::sumRegional ForData

4.27.3.42 vector<IFiles> getIFilesVector ( ) const [inline]

Definition at line 115 of file ModelData.h.

```
00115 {return iFilesVector;}
```

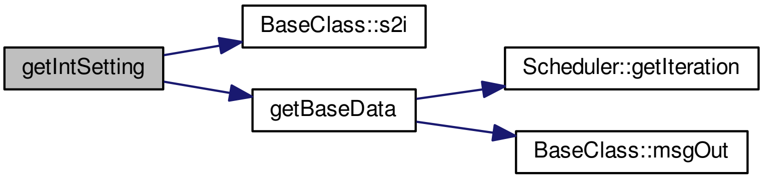
4.27.3.43 int getIntSetting (const string & *name_h,* int *position =* 0) const

Definition at line 998 of file ModelData.cpp.

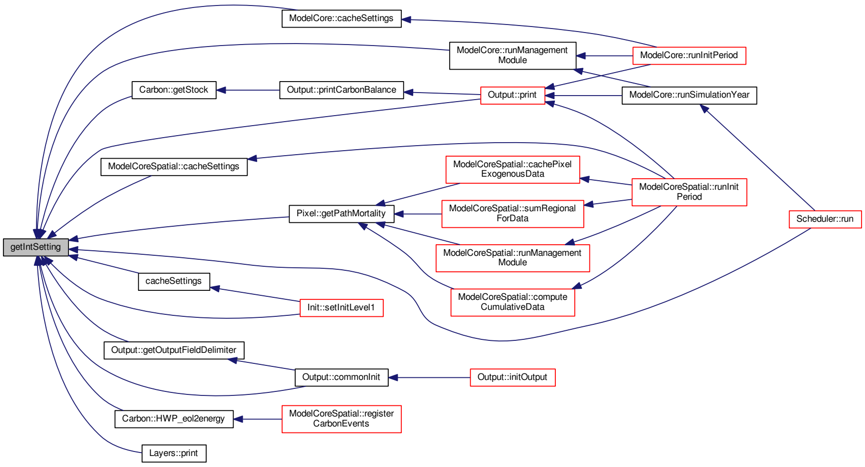
Referenced by ModelCore::cacheSettings(), ModelCoreSpatial::cacheSettings(), cacheSettings(), Output::commonInit(), Output::getOutputFieldDelimiter(), Pixel::getPathMortality(), Carbon::getStock(), Carbon::HWP_eol2energy(), Output::print(), Layers::print(), Scheduler::run(), ModelCore::runManagementModule(), and Init::setInitLevel1().

```
00998                                                         {
00999   return s2i( MTHREAD->MD->getBaseData(name_h,TYPE_INT,position) );
01000 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


ModelCore::cacheSettings
ModelCore::runManagement Module
ModelCore::runInitPeriod
Carbon::getStock
Output::printCarbonBalance
Output::print
ModelCore::runSimulationYear
ModelCoreSpatial::cacheSettings
ModelCoreSpatial::cachePixel ExogenousData
ModelCoreSpatial::runInit Period
Pixel::getPathMortality
ModelCoreSpatial::sumRegional ForData
Scheduler::run
getIntSetting
ModelCoreSpatial::runManagement Module
cacheSettings
ModelCoreSpatial::compute CumulativeData
Init::setInitLevel1
Output::getOutputFieldDelimiter
Output::commonInit
Output::initOutput
Carbon::HWP_eol2energy
ModelCoreSpatial::register CarbonEvents
Layers::print

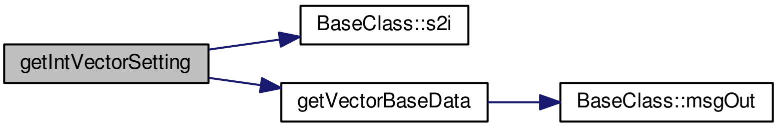
4.27.3.44 vector< int > getIntVectorSetting (const string & *name_h*) const

Definition at line 1014 of file ModelData.cpp.

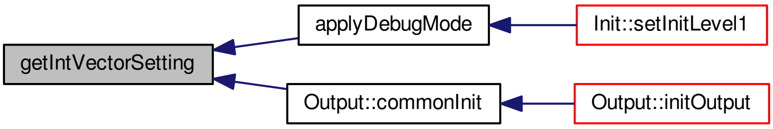
Referenced by applyDebugMode(), and Output::commonInit().

```
                                                       {
  return s2i(MTHREAD->MD->getVectorBaseData(name_h,
TYPE_INT));
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



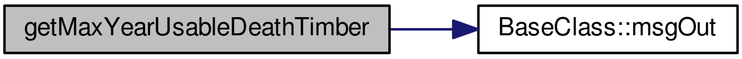
4.27.3.45 const int getMaxYearUsableDeathTimber (const string & *prod_h,* const string & *forType_h,* const string & *dClass_h*)

Definition at line 440 of file ModelData.cpp.

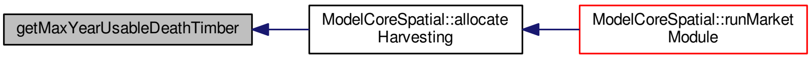
Referenced by ModelCoreSpatial::allocateHarvesting().

```

{
  for(uint i=0;i<forToProdVector.size();i++){
    if(    forToProdVector[i].product == prod_h
      && forToProdVector[i].forType == forType_h
      && forToProdVector[i].dClass  == dClass_h
    ){
      return forToProdVector[i].maxYears;
    }
  }
  msgOut(MSG_CRITICAL_ERROR,"In getMaxYearUsableDeathTimber() I has been asked of a
 combination that I don't know how to handle.");
}
```

Here is the call graph for this function:



Here is the caller graph for this function:


getMaxYearUsableDeathTimber
ModelCoreSpatial::allocate Harvesting
ModelCoreSpatial::runMarket Module

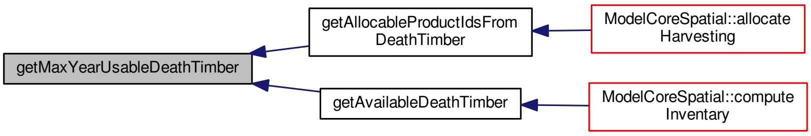
4.27.3.46 const int getMaxYearUsableDeathTimber ( )

Definition at line 428 of file ModelData.cpp.

Referenced by getAllocableProductIdsFromDeathTimber(), and getAvailableDeathTimber().

```
00428                                        {
00429   int maxMaxYears = 0;
00430   for(uint i=0;i<forToProdVector.size();i++){
00431     if(forToProdVector[i].maxYears > maxMaxYears){
00432       maxMaxYears = forToProdVector[i].maxYears;
00433     }
00434   }
00435   return maxMaxYears;
00436 }
```

Here is the caller graph for this function:


getMaxYearUsableDeathTimber
getAllocableProductIdsFrom DeathTimber
getAvailableDeathTimber
ModelCoreSpatial::allocate Harvesting
ModelCoreSpatial::compute Inventary

4.27.3.47 int getNForTypes ( ) [inline]

Definition at line 124 of file ModelData.h.

```
00124 {return forTypes.size();}
```

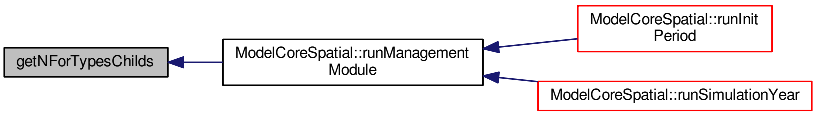
4.27.3.48 int getNForTypesChilds (const string & *forTypeId_h*)

Definition at line 133 of file ModelData.cpp.

Referenced by ModelCoreSpatial::runManagementModule().

```
00133                                                        {
00134     int nChilds = 0;
00135     for(int i=0;i<forTypes.size();i++){
00136         if(forTypes[i].ereditatedFrom==forTypeId_h) {
00137             nChilds ++;
00138         }
00139     }
00140     return nChilds;
00141 }
```

Here is the caller graph for this function:


getNForTypesChilds
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

4.27.3.49 int getNReclRules ( ) [inline]

Definition at line 125 of file ModelData.h.

```
00125 {return reclRules.size();}
```

4.27.3.50 string getOutputDirectory ( ) const [inline]

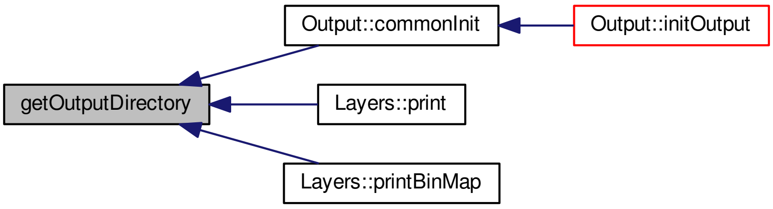
Return a vector of objects that together provide the specified resource in the specified quantity.

Definition at line 111 of file ModelData.h.

Referenced by Output::commonInit(), Layers::print(), and Layers::printBinMap().

```
00111 {return outputDirname;}
```

Here is the caller graph for this function:



4.27.3.51 vector< pathRule ∗ > getPathMortalityRule (const string & *forType,* const string & *dC*)

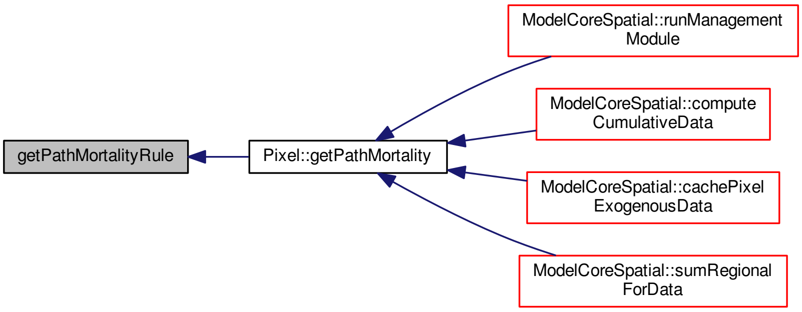
Return the pathogen mortality rule(s) associated with a given ft and dc (plural as more than a single pathogen could be found)

Definition at line 1890 of file ModelData.cpp.

Referenced by Pixel::getPathMortality().

```
01890                                                                    {
01891   vector<pathRule*> toReturn;
01892   for(uint i=0;i<pathRules.size();i++){
01893     if(pathRules[i].forType == forType && pathRules[i].dClass == dC){
01894       toReturn.push_back(&pathRules[i]);
01895     }
01896   }
01897   return toReturn;
01898 }
```

Here is the caller graph for this function:


ModelCoreSpatial::runManagement Module
ModelCoreSpatial::compute CumulativeData
getPathMortalityRule
Pixel::getPathMortality
ModelCoreSpatial::cachePixel ExogenousData
ModelCoreSpatial::sumRegional ForData

4.27.3.52 const double getProdData (const string & *type_h,* const int & *regId_h,* const string & *prodId_h,* const int & *year =* DATA_NOW*,* const string & *freeDim_h =* " ")

Basic function to retrieve products-related data. It addmits the following "filters": Name of the specific parameter requested Look for level1 or level 2 region. Product. It accept three keywords, for summing up all products, primary products or secondary products, namelly PROD_ALL, PROD_PRI, PROD_SEC. Unless specified, get the value of the current year. If array is smaller (e.g. because it is time-independent), get the last value. If specified, look exactly for it, otherwise simply doesn't filter for it.

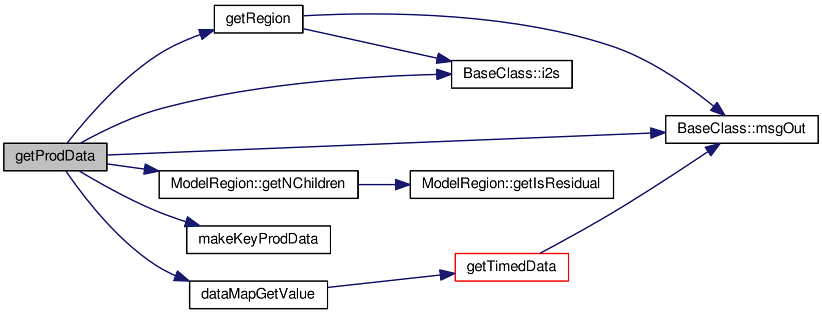
Definition at line 1108 of file ModelData.cpp.

Referenced by Carbon::getStock(), ModelCore::gpd(), ModelCoreSpatial::gpd(), Carbon::HWP_eol2energy(), Carbon::initialiseProductsStocks(), Output::printProductData(), and Carbon::registerProducts().

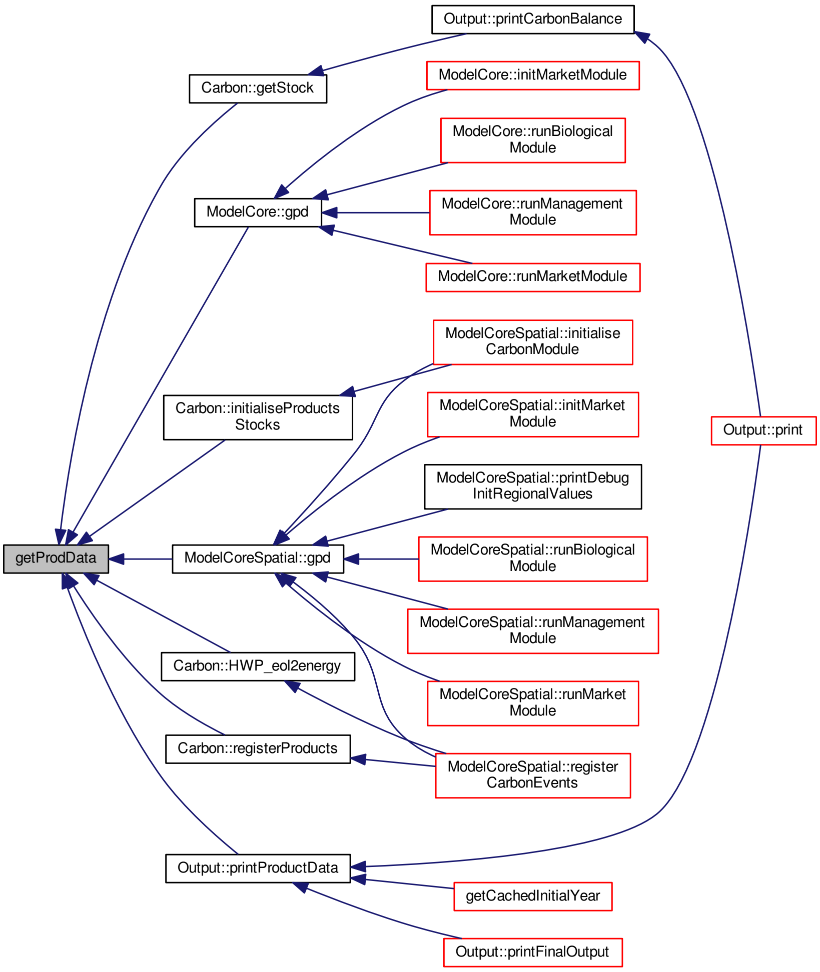
```
01108
                              {
01109
01110   double value=0;
01111   vector <int> regIds;
01112   string key;
01113   DataMap::const_iterator p;
01114
01115   bool found = false;
01116   vector <string> products;
```

```
  bool exactMatch=true;

  if(prodId_h == PROD_PRI){
    products = priProducts;
  } else if (prodId_h == PROD_SEC){
    products = secProducts;
  } else if (prodId_h == PROD_ALL || prodId_h == ""){
    products = allProducts;
    products.push_back("");
  } else  {
    products.push_back(prodId_h);
  }
  if(freeDim_h=="") exactMatch=false;

  // Make sure to set the new value to all l2 regions if requested for a reg1 level
  if(getRegion(regId_h)->getRegLevel()==2){
    regIds.push_back(regId_h);
  } else if (getRegion(regId_h)->getRegLevel()==1) {
    for(uint i=0;i<getRegion(regId_h)->getNChildren();i++){
      regIds.push_back(getRegion(regId_h)->getChildren()[i]->getRegId());
    }
  } else {
    msgOut(MSG_CRITICAL_ERROR, "Error in setProdData(). Setting a value for the
 whole World is not supported.");
  }
  int regIdsS = regIds.size();


  for(uint r=0;r<regIdsS;r++){
    for(uint i=0;i<products.size();i++){
      key = makeKeyProdData(type_h,i2s(regIds[r]),products[i],freeDim_h);
      if (!exactMatch && key.size () > 0)  key.resize (key.size () - 1); // bug 20140402, removing the last
 #
      value += dataMapGetValue(prodDataMap,key,year,exactMatch);
      if(tempBool) found = true;
    }
  }

  if(!found){
    msgOut(errorLevel, "Error in getProdData: no combination found for "+type_h+", "+
i2s(regId_h)+", "+prodId_h+", "+i2s(year)+", "+freeDim_h+". Returning 0, but double check that this
 is ok for your model.");
  }
  return value;


}
```

Here is the call graph for this function:



Here is the caller graph for this function:



Output::printCarbonBalance
Carbon::getStock
ModelCore::initMarketModule
ModelCore::runBiological Module
ModelCore::gpd
ModelCore::runManagement Module
ModelCore::runMarketModule
ModelCoreSpatial::initialise CarbonModule
Carbon::initialiseProducts Stocks
ModelCoreSpatial::initMarket Module
Output::print
ModelCoreSpatial::printDebug InitRegionalValues
getProdData
ModelCoreSpatial::gpd
ModelCoreSpatial::runBiological Module
ModelCoreSpatial::runManagement Module
Carbon::HWP_eol2energy
ModelCoreSpatial::runMarket Module
Carbon::registerProducts
ModelCoreSpatial::register CarbonEvents
Output::printProductData
getCachedInitialYear
Output::printFinalOutput

4.27.3.53 reclRule∗ getReclRule (int *position*) [inline]

Definition at line 135 of file ModelData.h.

```
00135 {return &reclRules[position];}
```

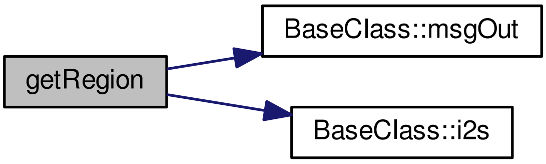
4.27.3.54 ModelRegion ∗ getRegion (int *regId_h*)

Definition at line 318 of file ModelData.cpp.

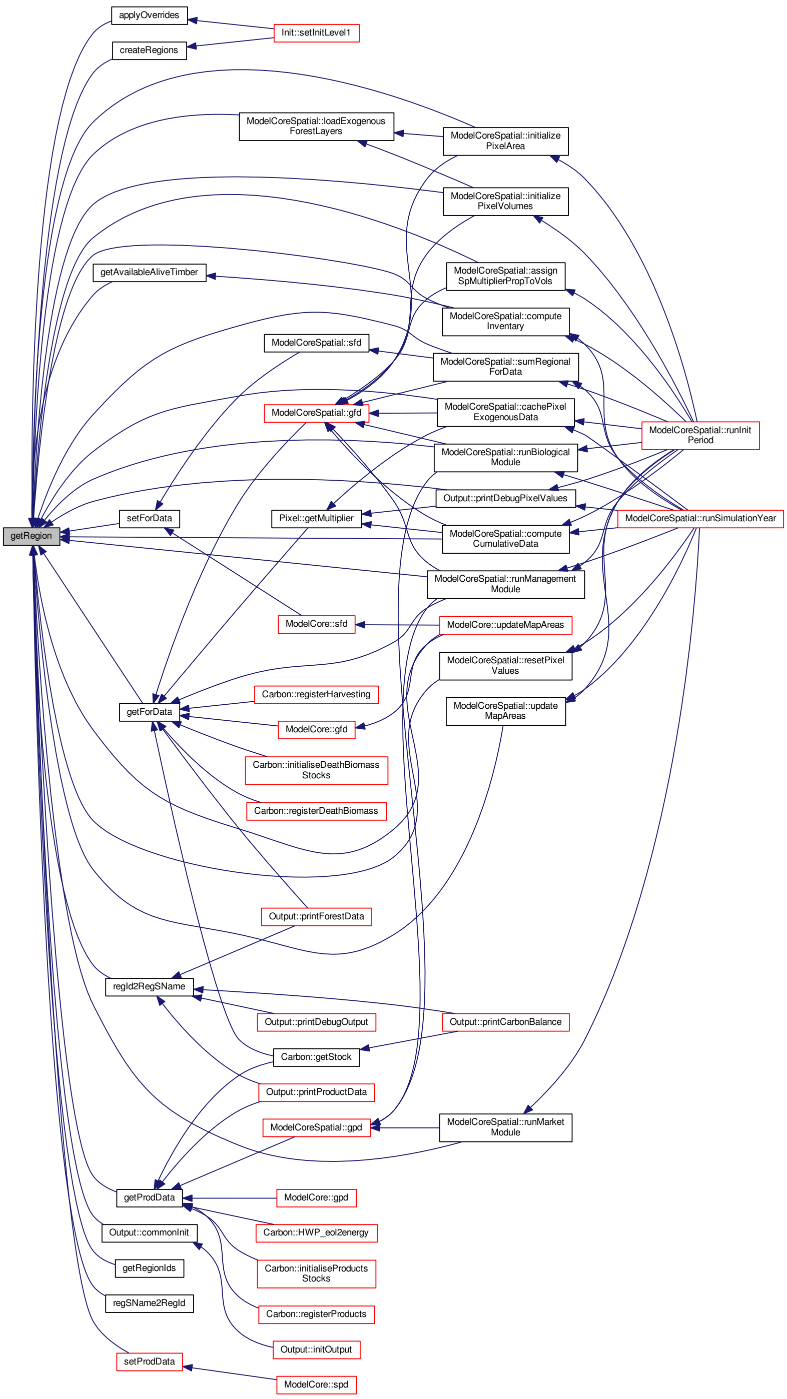
Referenced by applyOverrides(), ModelCoreSpatial::assignSpMultiplierPropToVols(), ModelCoreSpatial::cachePixelExogenousData(), Output::commonInit(), ModelCoreSpatial::computeCumulativeData(), ModelCoreSpatial::computeInventary(), createRegions(), getAvailableAliveTimber(), getForData(), getProdData(), getRegionIds(), ModelCoreSpatial::initializePixelArea(), ModelCoreSpatial::initializePixelVolumes(), ModelCoreSpatial::loadExogenousForestLayers(), Output::printDebugPixelValues(), regId2RegSName(), regSName2RegId(), ModelCoreSpatial::resetPixelValues(), ModelCoreSpatial::runBiologicalModule(), ModelCoreSpatial::runManagementModule(), ModelCoreSpatial::runMarketModule(), setForData(), setProdData(), ModelCoreSpatial::sumRegionalForData(), ModelCore::updateMapAreas(), and ModelCoreSpatial::updateMapAreas().

```
                                {
  for (int i=0; i< regionsVector.size();i++){
    if(regionsVector[i].getRegId()==regId_h){
      return &regionsVector[i];
    }
  }
  msgOut(MSG_CRITICAL_ERROR, "Region id "+i2s(regId_h)+" not found, check your input data. Aborting simulation.");
}
```

Here is the call graph for this function:



Here is the caller graph for this function:


applyOverrides
Init::setInitLevel1
createRegions
ModelCoreSpatial::loadExogenous ForestLayers
ModelCoreSpatial::initialize PixelArea
ModelCoreSpatial::initialize PixelVolumes
getAvailableAliveTimber
ModelCoreSpatial::assign SpMultiplierPropToVols
ModelCoreSpatial::compute Inventory
ModelCoreSpatial::sfd
ModelCoreSpatial::sumRegional ForData
ModelCoreSpatial::gfd
ModelCoreSpatial::cachePixel ExogenousData
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runBiological Module
Output::printDebugPixelValues
getRegion
setForData
Pixel::getMultiplier
ModelCoreSpatial::runSimulationYear
ModelCoreSpatial::compute CumulativeData
ModelCoreSpatial::runManagement Module
ModelCore::sfd
ModelCore::updateMapAreas
ModelCoreSpatial::resetPixel Values
Carbon::registerHarvesting
getForData
ModelCore::gfd
ModelCoreSpatial::update MapAreas
Carbon::initialiseDeathBiomass Stocks
Carbon::registerDeathBiomass
Output::printForestData
regId2RegSName
Output::printDebugOutput
Output::printCarbonBalance
Carbon::getStock
Output::printProductData
ModelCoreSpatial::gpd
ModelCoreSpatial::runMarket Module
getProdData
ModelCore::gpd
Output::commonInit
Carbon::HWP_eol2energy
getRegionIds
Carbon::initialiseProducts Stocks
regSName2RegId
Carbon::registerProducts
Output::initOutput
setProdData
ModelCore::spd

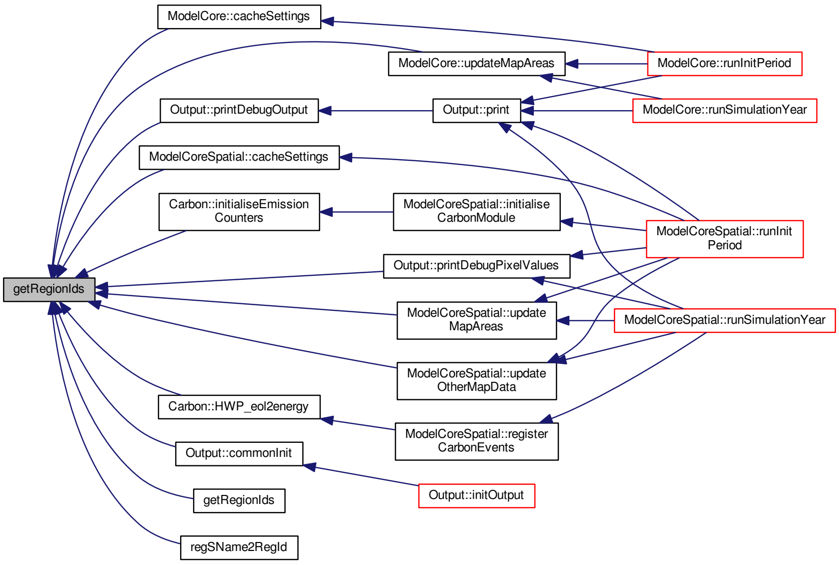
4.27.3.55 vector< int > getRegionIds (int *level_h*, bool *excludeResidual* = `true`)

Definition at line 338 of file ModelData.cpp.

Referenced by ModelCore::cacheSettings(), ModelCoreSpatial::cacheSettings(), Output::commonInit(), getRegionIds(), Carbon::HWP_eol2energy(), Carbon::initialiseEmissionCounters(), Output::printDebugOutput(), Output::printDebugPixelValues(), regSName2RegId(), ModelCore::updateMapAreas(), ModelCoreSpatial::updateMapAreas(), and ModelCoreSpatial::updateOtherMapData().

```
00338                                                       {
00339   vector <int> toReturn;
00340   for(uint i=0;i<regionsVector.size();i++){
00341     if(regionsVector[i].getRegLevel()==level_h){
00342       if( (!excludeResidual) || (!regionsVector[i].getIsResidual())){
00343         toReturn.push_back(regionsVector[i].getRegId());
00344       }
00345     }
00346   }
00347   return toReturn;
00348 }
```

Here is the caller graph for this function:

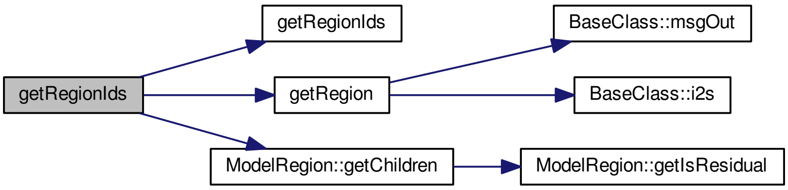

ModelCore::cacheSettings
ModelCore::updateMapAreas
ModelCore::runInitPeriod
Output::printDebugOutput
Output::print
ModelCore::runSimulationYear
ModelCoreSpatial::cacheSettings
Carbon::initialiseEmission Counters
ModelCoreSpatial::initialise CarbonModule
ModelCoreSpatial::runInit Period
Output::printDebugPixelValues
getRegionIds
ModelCoreSpatial::update MapAreas
ModelCoreSpatial::runSimulationYear
ModelCoreSpatial::update OtherMapData
Carbon::HWP_eol2energy
ModelCoreSpatial::register CarbonEvents
Output::commonInit
Output::initOutput
getRegionIds
regSName2RegId

4.27.3.56 vector< vector< int > > getRegionIds (bool *excludeResidual* = `true`)

Definition at line 362 of file ModelData.cpp.

```
00362                                                 {
00363   vector < vector <int> > toReturn;
00364   vector <int> l1regIds =  MTHREAD->MD->getRegionIds(1, excludeResidual);
00365   for(uint i=0;i<l1regIds.size();i++){
00366     vector<int> l2ChildrenIds;
00367     ModelRegion* l1Region = MTHREAD->MD->getRegion(l1regIds[i]);
00368     vector<ModelRegion*> l2Childrens = l1Region->getChildren(excludeResidual);
00369     for(uint j=0;j<l2Childrens.size();j++){
00370       l2ChildrenIds.push_back(l2Childrens[j]->getRegId());
00371     }
00372     if(l2ChildrenIds.size()){
00373       toReturn.push_back(l2ChildrenIds);
00374     }
00375   }
00376   return toReturn;
00377 }
```

Here is the call graph for this function:



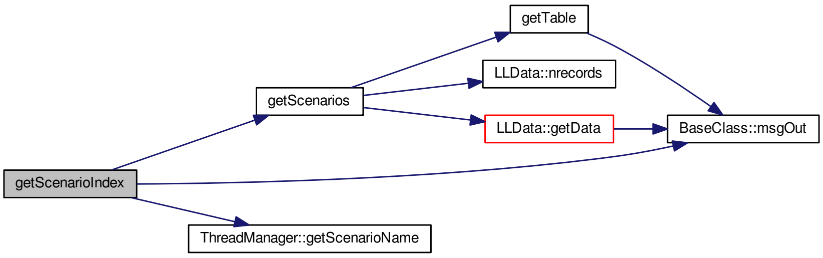
4.27.3.57 int getScenarioIndex ( )

Todo Check that I can call this function all around the model and not only at the beginning

Definition at line 155 of file ModelData.cpp.

```
00155                             {
00156   vector<string> scenarios = getScenarios(); /// \todo Check that I can call this
       function all around the model and not only at the beginning
00157   string currentScenario = MTHREAD->getScenarioName();
00158   for(int i=0;i<scenarios.size();i++){
00159     if (currentScenario == scenarios[i]){
00160       return i;
00161     }
00162   }
00163   msgOut(MSG_CRITICAL_ERROR, "function getScenarioIndex() didn't found the current
       scenarioName within those returned by getScenarios().");
00164   return 0;
00165 }
```

Here is the call graph for this function:



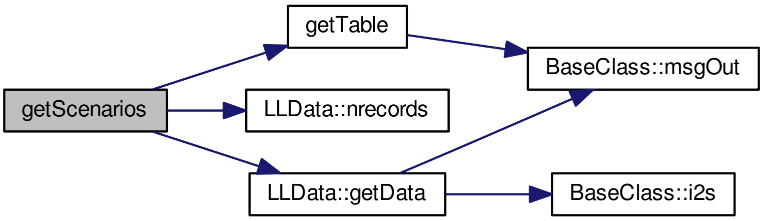
4.27.3.58 vector< string > getScenarios ( )

Definition at line 144 of file ModelData.cpp.

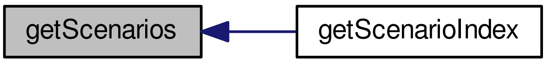
Referenced by getScenarioIndex().

```
00144                        {
00145   vector<string> toReturn;
00146   LLData table = getTable("scenarios");
00147   for(int i=0;i<table.nrecords();i++){
00148     string scenarioName = table.getData(i,"id");
00149     toReturn.push_back(scenarioName);
00150   }
00151   return toReturn;
00152 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



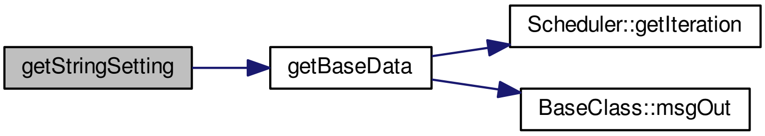
4.27.3.59 string getStringSetting (const string & *name_h*, int *position* = 0) const

Definition at line 1006 of file ModelData.cpp.

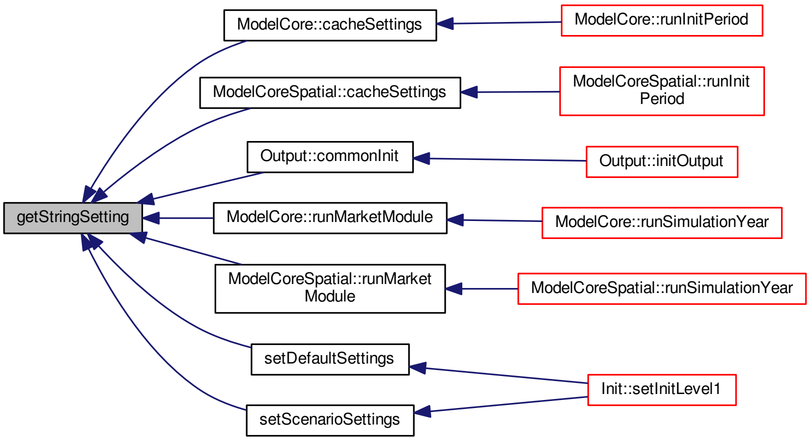
Referenced by ModelCore::cacheSettings(), ModelCoreSpatial::cacheSettings(), Output::commonInit(), ModelCore::runMarketModule(), ModelCoreSpatial::runMarketModule(), setDefaultSettings(), and setScenarioSettings().

```
01006                                                               {
01007   return MTHREAD->MD->getBaseData(name_h,TYPE_STRING,position);
01008 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


getStringSetting
ModelCore::cacheSettings
ModelCoreSpatial::cacheSettings
Output::commonInit
ModelCore::runMarketModule
ModelCoreSpatial::runMarket Module
setDefaultSettings
setScenarioSettings
ModelCore::runInitPeriod
ModelCoreSpatial::runInit Period
Output::initOutput
ModelCore::runSimulationYear
ModelCoreSpatial::runSimulationYear
Init::setInitLevel1

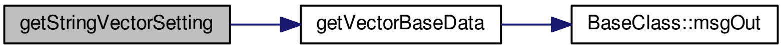
4.27.3.60 vector< string > getStringVectorSetting (const string & *name_h*) const

Definition at line 1022 of file ModelData.cpp.

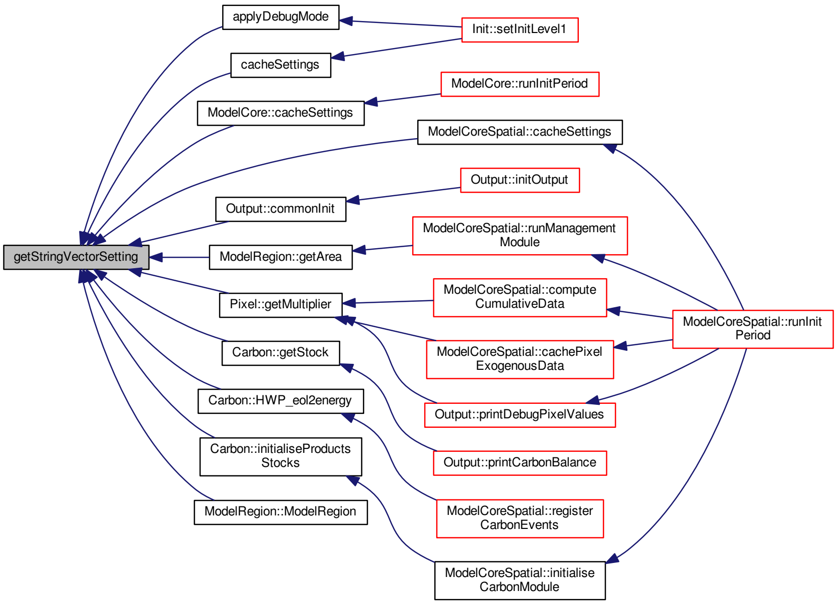
Referenced by applyDebugMode(), ModelCore::cacheSettings(), ModelCoreSpatial::cacheSettings(), cacheSettings(), Output::commonInit(), ModelRegion::getArea(), Pixel::getMultiplier(), Carbon::getStock(), Carbon::HWP_eol2energy(), Carbon::initialiseProductsStocks(), and ModelRegion::ModelRegion().

```
01022                                                           {
01023   return MTHREAD->MD->getVectorBaseData(name_h,
      TYPE_STRING);
01024 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


applyDebugMode
Init::setInitLevel1
cacheSettings
ModelCore::runInitPeriod
ModelCore::cacheSettings
ModelCoreSpatial::cacheSettings
Output::initOutput
Output::commonInit
ModelCoreSpatial::runManagement Module
getStringVectorSetting
ModelRegion::getArea
ModelCoreSpatial::compute CumulativeData
Pixel::getMultiplier
ModelCoreSpatial::runInit Period
ModelCoreSpatial::cachePixel ExogenousData
Carbon::getStock
Carbon::HWP_eol2energy
Output::printDebugPixelValues
Carbon::initialiseProducts Stocks
Output::printCarbonBalance
ModelRegion::ModelRegion
ModelCoreSpatial::register CarbonEvents
ModelCoreSpatial::initialise CarbonModule

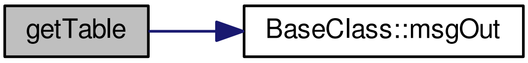
4.27.3.61 LLData getTable (string *tableName_h,* int *debugLevel* = MSG_CRITICAL_ERROR)

Definition at line 1657 of file ModelData.cpp.

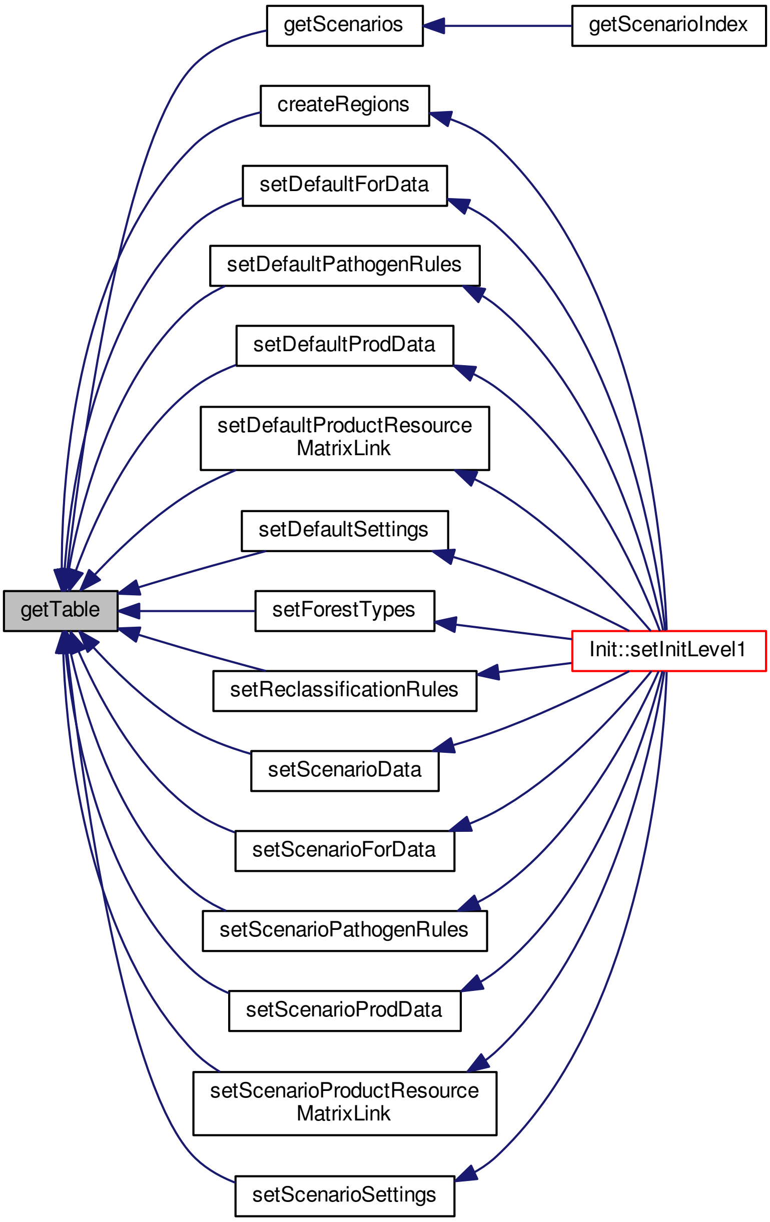
Referenced by createRegions(), getScenarios(), setDefaultForData(), setDefaultPathogenRules(), setDefaultProdData(), setDefaultProductResourceMatrixLink(), setDefaultSettings(), setForestTypes(), setReclassificationRules(), setScenarioData(), setScenarioForData(), setScenarioPathogenRules(), setScenarioProdData(), setScenarioProductResourceMatrixLink(), and setScenarioSettings().

```
01657                                                      {
01658   LLData toReturn(MTHREAD,"");
01659   for(uint i=0;i<LLDataVector.size();i++){
01660     if (LLDataVector[i].getTableName() == tableName_h)return
      LLDataVector[i];
01661   }
01662   msgOut(debugLevel,"No table found with name "+tableName_h);
01663   return toReturn;
01664 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.3.62 bool getTempBool ( ) [inline]

Definition at line 142 of file ModelData.h.

```
00142 {return tempBool;}
```

4.27.3.63 double getTimedData (const vector< double > & *dated_vector,* const int & *year_h*) const

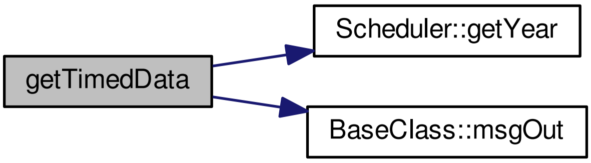
Return the value for the specified year in a timelly ordered vector, taking the last value if this is smaller than the required position.

Definition at line 1370 of file ModelData.cpp.

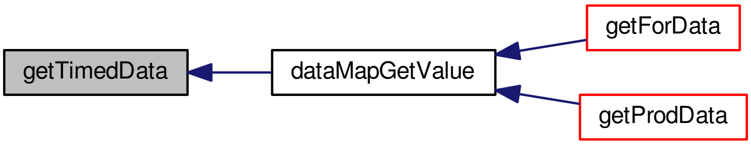
Referenced by dataMapGetValue().

```
                                                                                    {

  int position;
  if(year_h==DATA_NOW){
    position = MTHREAD->SCD->getYear()-cached_initialYear;
  } else {
    position = year_h-cached_initialYear;
  }

  if(dated_vector.size() > position) {
    return dated_vector[position];
  } else if (dated_vector.size() > 0 ){
    // returning the last available value...
    return dated_vector[dated_vector.size()-1];
  } else {
    msgOut(MSG_CRITICAL_ERROR, "Error in getTimedData: requested value doesn't have
any value, even on the first position(year)!");
  }
  return 0;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



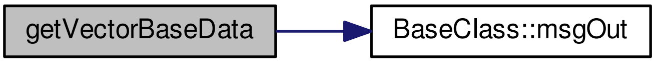
4.27.3.64 vector< string > getVectorBaseData (const string & *name_h*, int *type_h*) [private]

Definition at line 983 of file ModelData.cpp.

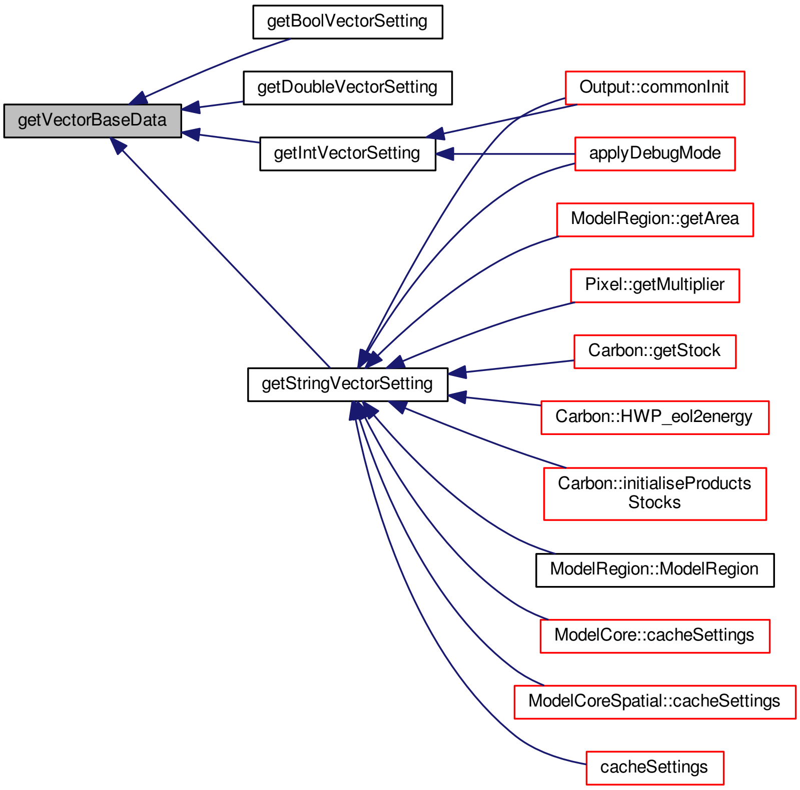
Referenced by getBoolVectorSetting(), getDoubleVectorSetting(), getIntVectorSetting(), and getStringVectorSetting().

```
00983                                                              {
00984   for (uint i=0; i<programSettingsVector.size();i++){
00985     if (programSettingsVector.at(i).name == name_h){
00986       int type = programSettingsVector.at(i).type;
00987       if(type != type_h){msgOut(MSG_CRITICAL_ERROR, "mismatching type in calling
       getVectorBaseData() for "+name_h);}
00988       return programSettingsVector.at(i).values;
00989     }
00990   }
00991   msgOut(MSG_CRITICAL_ERROR, "Error calling getVectorBaseData() for "+ name_h +".
       No setting option or macro data found with this name.");
00992   vector <string> toReturn;
00993   return toReturn;
00994 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.3.65 void loadDataFromCache (string *tablename*)

Load data from a cached CSV instead of the openoffice file.

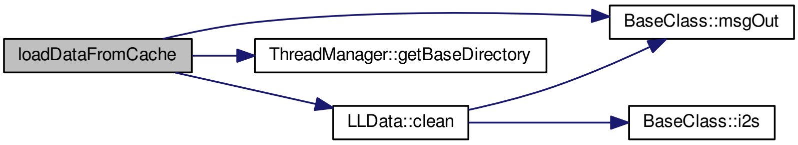
Definition at line 1589 of file ModelData.cpp.

Referenced by loadInput(), and LLData::nheaders().

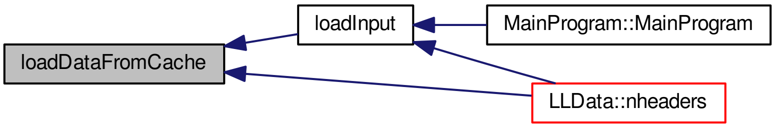
```
                                                   {
  msgOut(MSG_INFO,"Attention, using cached data (csv) for "+tablename);
  string fileName = MTHREAD->getBaseDirectory()+"cachedInput/"+tablename+".csv";
  QFile file(fileName.c_str());
  if (!file.open(QFile::ReadOnly)) {
      msgOut(MSG_ERROR, "Cannot open cached file "+fileName+" for reading.");
  }
  QTextStream in(&file);
  LLData data(MTHREAD, tablename);
  int countRow = 0;
  while (!in.atEnd()) {
    QString line = in.readLine();
    QStringList fields = line.split(';');
    if (countRow==0){ // building headers
      for(uint i =0;i<fields.size();i++){
        data.headers.push_back(fields.at(i).toStdString());
      }
    } else {
      vector<string> record ; //= fields.toVector().toStdVector();
      unsigned int z = fields[0].toStdString().find("#");
      if( z!=string::npos && z == 0) continue; // found "#" on fist position, it's a comment!
      for(uint i =0;i<fields.size();i++){
        string field = fields.at(i).toStdString();
```

```
01612         replace(field.begin(), field.end(), ',', '.');
01613         record.push_back(field);
01614       }
01615       data.records.push_back(record);
01616     }
01617     countRow++;
01618   }
01619   data.clean();
01620   LLDataVector.push_back(data);
01621
01622 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.3.66 void loadInput ( )

Unzip the OpenOffice input file (NEW 2008.05.13)

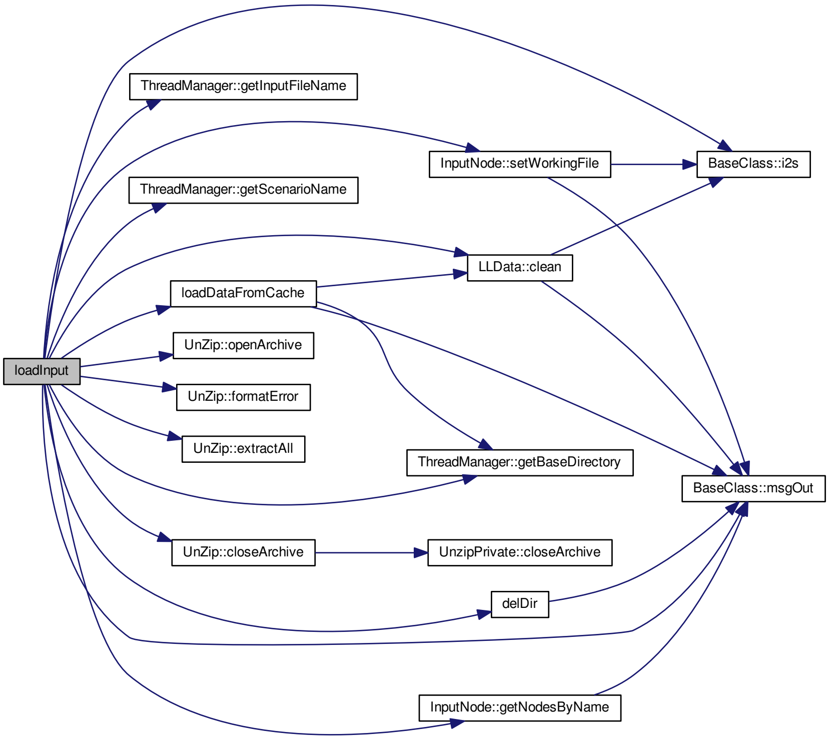
Definition at line 1417 of file ModelData.cpp.

Referenced by MainProgram::MainProgram(), and LLData::nheaders().

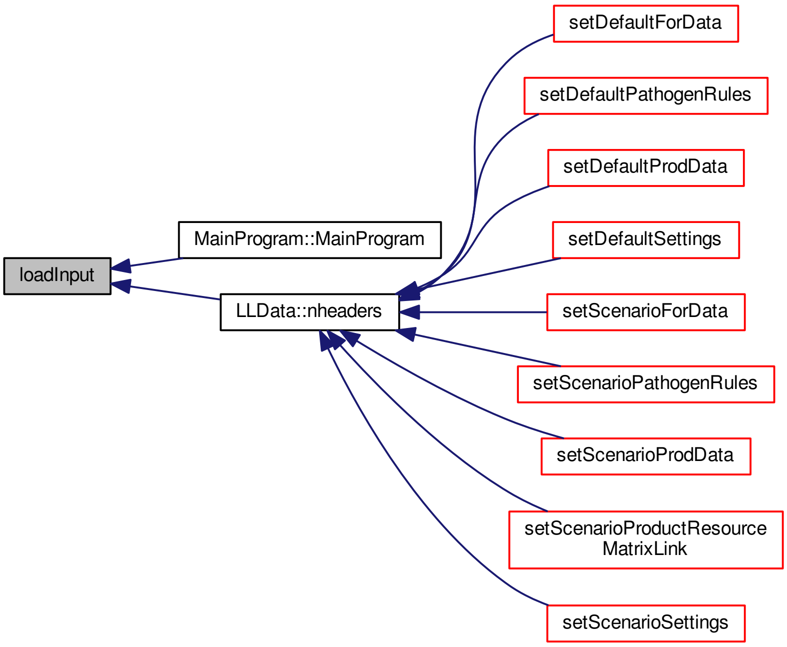
```
01417                 {
01418   msgOut(MSG_INFO, "Loading input files (this can take a few minutes)...");
01419   //QString iFile("data/ffsmInput.ods");
01420   QString iFile(MTHREAD->getInputFileName().c_str());
01421   //cout << "PIPPO !!!!! " << MTHREAD->getInputFileName().c_str() << endl;
01422
01423   //std::random_device rd;
01424   //std::mt19937 localgen(rd());
01425   std::mt19937 localgen(time(0));
01426   std::uniform_int_distribution<> dis(10, 1000000);
01427   int randomNumber = dis(localgen);
01428
01429   QString oDir((MTHREAD->getBaseDirectory()+"tempInput-"+
      MTHREAD->getScenarioName()+i2s(randomNumber)).c_str());
01430   string forDataCachedFilename = MTHREAD->getBaseDirectory()+"
      cachedInput/forData.csv";
01431   string prodDataCachedFilename = MTHREAD->getBaseDirectory()+"
      cachedInput/prodData.csv";
```

```

  // removing output directory if exist..
  QDir oQtDir(oDir);

  if(oQtDir.exists()){
    bool deleted;
    deleted = delDir(oDir);
    if(deleted){msgOut(MSG_DEBUG,"Correctly deleted old temporary data");}
    else {msgOut(MSG_WARNING, "I could not delete old temporary data dir (hopefully we’ll
 overrite the input files)");}
  }

  if (!QFile::exists(iFile))
  {
    cout << "File does not exist." << endl << endl;
    //return false;
  }
  UnZip::ErrorCode ec;
  UnZip uz;
  ec = uz.openArchive(iFile);
  if (ec != UnZip::Ok)  {
        //cout << "Failed to open archive: " << uz.formatError(ec).toAscii().data() << endl << endl; // Qt4
        cout << "Failed to open archive: " << uz.formatError(ec).toLatin1().data() << endl <<
endl;  // Qt5
    //return false;
  }
  ec = uz.extractAll(oDir);
  if (ec != UnZip::Ok){
    //cout << "Extraction failed: " << uz.formatError(ec).toAscii().data() << endl << endl; // Qt4
    cout << "Extraction failed: " << uz.formatError(ec).toLatin1().data() << endl << endl;  //
 Qt5
    uz.closeArchive();
    //return false;
  }

  // loading input file into memory...
  string inputXMLFileName = MTHREAD->getBaseDirectory()+"tempInput-"+
MTHREAD->getScenarioName()+i2s(randomNumber)+"/content.xml";
  //string inputXMLFileName = MTHREAD->getBaseDirectory()+"test/content.xml";
  //cout << "inputXMLFileName: " << inputXMLFileName << endl;
  //mainDocument = new InputDocument();
  mainDocument.setWorkingFile(inputXMLFileName);
  //InputNode documentContent = mainDocument.getNodeByName("office:document-content");
  //InputNode documentBody = mainDocument.getNodeByName("office:body");
  //InputNode mainNode = mainDocument.getNodeByName("spreadsheet");
  //InputNode pippo = mainDocument.getNodeByName("pippo-pippo");
  //InputNode table = mainDocument.getNodeByName("table");
  //cout << "Test result: " << table.getStringContent() << endl;


  vector <InputNode> tables = mainDocument.getNodesByName("table");
  for(uint i=0;i<tables.size();i++){
    string tableName = tables[i].getStringAttributeByName("name");
    //cout <<tableName<<endl;
    if( tableName == "forData"){
      if(QFile::exists(forDataCachedFilename.c_str())){
        loadDataFromCache("forData");
        continue;
      }
    } else if (tableName == "prodData"){
        if (QFile::exists(prodDataCachedFilename.c_str())) {
          loadDataFromCache("prodData");
        continue;
        }
    }
    LLData data(MTHREAD,tables[i].getStringAttributeByName("name"));
    vector <InputNode> rows = tables[i].getNodesByName("table-row",MSG_NO_MSG,true);
    if(rows.size()<2) continue; //empty table or only with headers
    // building headers..
    vector <InputNode> cells = rows[0].getNodesByName("table-cell",MSG_NO_MSG,true);
    for (uint y=0; y<cells.size(); y++){
      int repeated = 1;
      if( cells[y].hasAttributeByName("number-columns-repeated")){
        repeated = cells[y].getIntAttributeByName("number-columns-repeated");
      }
      for (int q=0;q<repeated;q++){
        if( !cells[y].hasChildNode("p") ){
          data.headers.push_back(""); // empty header
        } else {
          data.headers.push_back(cells[y].getNodeByName("p",MSG_NO_MSG,true).getStringContent());
        }
      }
    }
    // loading data...
    for (uint j=1; j<rows.size();j++){
      //cout << j << endl;
      vector <InputNode> cells = rows[j].getNodesByName("table-cell",MSG_NO_MSG,true);
      //vector <InputNode> cells = rows[j].getChildNodes();
      if (cells.size()<1) continue;
      vector<string> record;
      // checking the first cell is not a comment nor is empty..
      int childCount = cells[0].getChildNodesCount();
      if (childCount == 0 || !cells[0].hasChildNode("p")) continue; // empty line, first column empty!
      string fistCol = cells[0].getNodeByName("p",MSG_NO_MSG,true).getStringContent();
      unsigned int z;
      z = fistCol.find("#");
      if( z!=string::npos && z == 0) continue; // found "#" on fist position, it's a comment!
      for (uint y=0; y<cells.size(); y++){
        int repeated = 1;
        if( cells[y].hasAttributeByName("number-columns-repeated")){
          repeated = cells[y].getIntAttributeByName("number-columns-repeated");
        }
        for (int q=0;q<repeated;q++){
          if( !cells[y].hasChildNode("p") ){
            record.push_back(""); // empty header
          } else {
            // changed 20120625 as for float values the content of p is the visualised value, not the full memorised one.
            // this is strange because  tought I already tested it.. but maybe is changed the format??
            if(cells[y].getStringAttributeByName("value-type")=="float"){
              record.push_back(cells[y].getStringAttributeByName("value"));
            } else {
              record.push_back(cells[y].getNodeByName("p",MSG_NO_MSG,true).getStringContent());
            }
          }
        }
      }
      data.records.push_back(record);
    }
    data.clean();
    LLDataVector.push_back(data);
  }

  //debug !!!
  /*for (uint i=0; i<LLDataVector.size();i++){
    cout << "***************** NEW TABLE: " << LLDataVector[i].tableName << endl;
    //cout << "***** Headers: "<< endl;
    int headerSize = LLDataVector[i].headers.size();
    bool ok = true;
    cout << "Header size: " << headerSize << endl;
    //for (uint j=0; j<LLDataVector[i].headers.size();j++){
    //  cout << "["<<j<<"] " << LLDataVector[i].headers[j] << endl;
    //}
    //cout << "***** Records: " << endl;
    for (uint j=0; j<LLDataVector[i].records.size();j++){
      //cout << "** Record "<<j<<":"<<endl;
      if(LLDataVector[i].records[j].size() != headerSize){
        cout << "There is a problem on record " << j <<"!"<< endl;
        cout << "His size is: "<< LLDataVector[i].records[j].size() << endl;
        ok = false;
      }
      //for (uint y=0; y<LLDataVector[i].records[j].size();y++){
      //  cout << "["<<y<<"] " << LLDataVector[i].records[j][y] << endl;
      //}
    }
    if(!ok) {cout <<"Problems with this table :-( !"<<endl;}
  }*/



  // deleting output directory if exist...
  if(oQtDir.exists()){
    bool deleted;
    deleted = delDir(oDir);
    if(deleted){msgOut(MSG_DEBUG,"Correctly deleted old temporary data");}
    else {msgOut(MSG_WARNING, "I could not delete old temporary data dir (hopefully we'll overrite the input files)");}
  }
}
```

Here is the call graph for this function:


loadInput
ThreadManager::getInputFileName
InputNode::setWorkingFile
BaseClass::i2s
ThreadManager::getScenarioName
LLData::clean
loadDataFromCache
UnZip::openArchive
UnZip::formatError
UnZip::extractAll
ThreadManager::getBaseDirectory
BaseClass::msgOut
UnZip::closeArchive
UnzipPrivate::closeArchive
delDir
InputNode::getNodesByName

Here is the caller graph for this function:



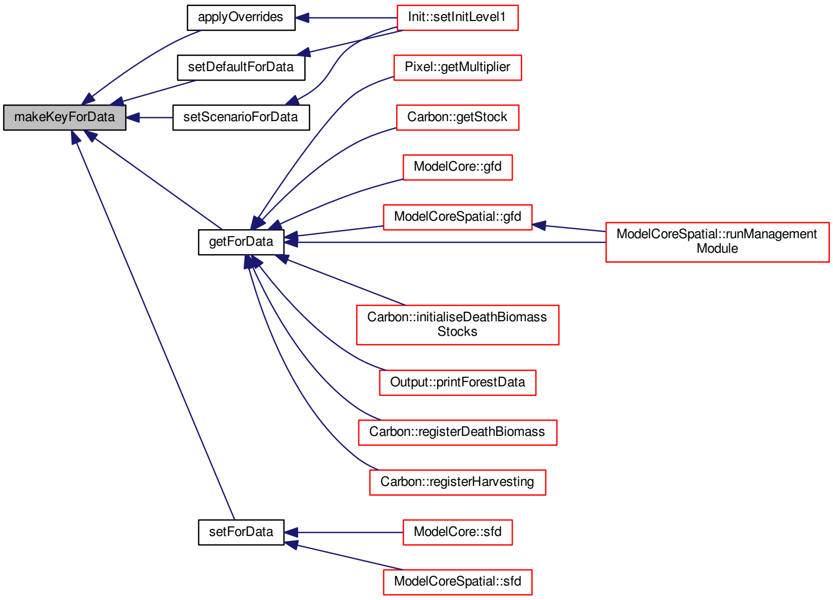
4.27.3.67 string makeKeyForData (const string & *parName*, const string & *regId*, const string & *forType*, const string & *diamClass*) const [inline]

Definition at line 166 of file ModelData.h.

Referenced by applyOverrides(), getForData(), setDefaultForData(), setForData(), and setScenarioForData().

```
00166 {return parName+"#"+regId+"#"+forType+"#"+diamClass+"#";}
```

Here is the caller graph for this function:



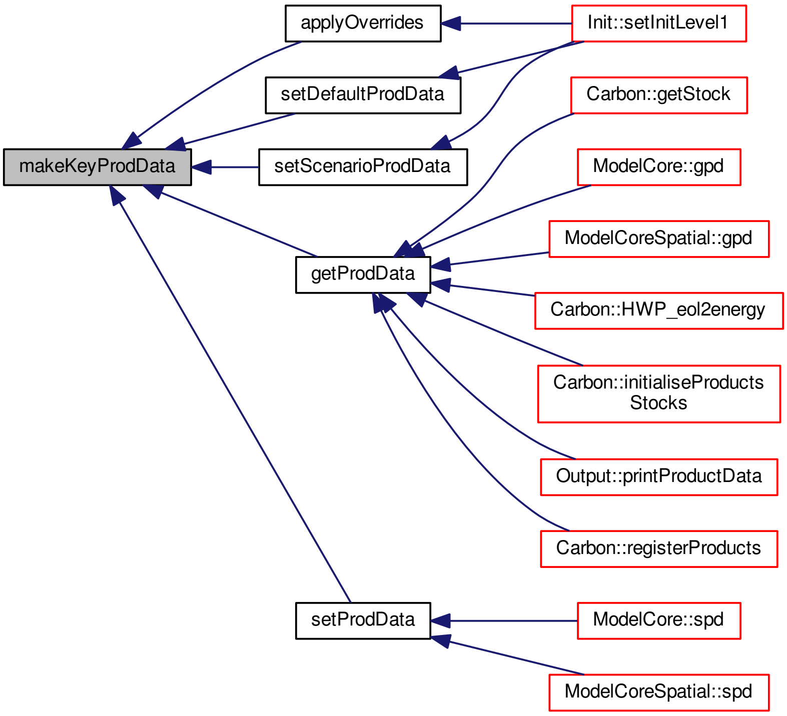
4.27.3.68 string makeKeyProdData (const string & *parName*, const string & *regId*, const string & *prod*, const string & *freeDim* = "") const [inline]

Definition at line 165 of file ModelData.h.

Referenced by applyOverrides(), getProdData(), setDefaultProdData(), setProdData(), and setScenarioProdData().

```
00165 {return parName+"#"+regId+"#"+prod+"#"+freeDim+"#";}
```

Here is the caller graph for this function:



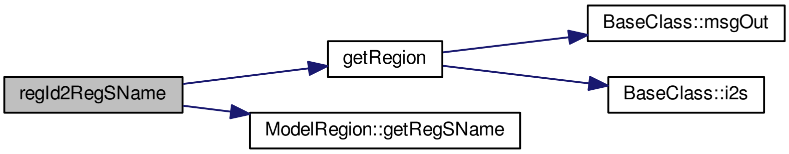
4.27.3.69 string regId2RegSName (const int & *regId_h*) const

Definition at line 380 of file ModelData.cpp.

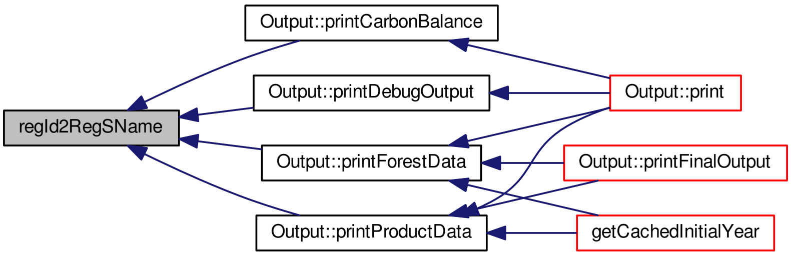
Referenced by Output::printCarbonBalance(), Output::printDebugOutput(), Output::printForestData(), and Output::printProductData().

```
00380                                              {
00381   ModelRegion* reg = MTHREAD->MD->getRegion(regId_h);
00382   return reg->getRegSName();
00383 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.3.70 bool regionExist (const int & *regId_h*) const

Definition at line 328 of file ModelData.cpp.

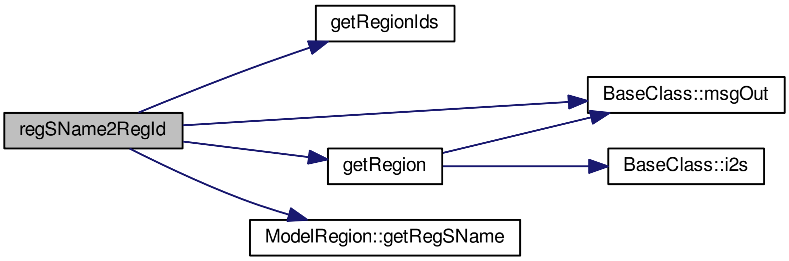
```
00328                                                    {
00329   for (int i=0; i< regionsVector.size();i++){
00330     if(regionsVector[i].getRegId()==regId_h){
00331       return true;
00332     }
00333   }
00334   return false;
00335 }
```

4.27.3.71 int regSName2RegId (const string & *regSName_h*) const

Definition at line 386 of file ModelData.cpp.

```
00386                                                            {
00387   ModelRegion* reg;
00388   for(uint i=0; i<3; i++){
00389     vector <int> regIds =  MTHREAD->MD->getRegionIds(i, false);
00390     for(uint j=0;j<regIds.size();j++){
00391       reg = MTHREAD->MD->getRegion(regIds[j]);
00392       if(reg->getRegSName()==regSName_h) {return regIds[j];}
00393     }
00394   }
00395   msgOut(MSG_CRITICAL_ERROR,"Regional short name not found.");
00396 }
```

Here is the call graph for this function:



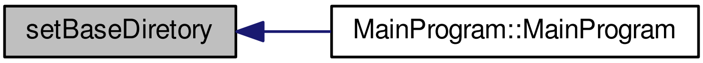
4.27.3.72 void setBaseDiretory (string *baseDirectory_h*) [inline]

Definition at line 176 of file ModelData.h.

Referenced by MainProgram::MainProgram().

```
00176 {baseDirectory=baseDirectory_h;}
```

Here is the caller graph for this function:



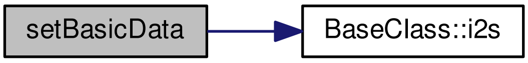
4.27.3.73 void setBasicData (const string & *name_h,* int *value,* int *position* = 0)

Definition at line 1033 of file ModelData.cpp.

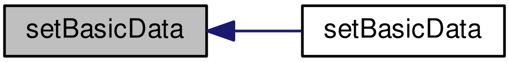
Referenced by setBasicData().

```
01033                                                                {
01034   setBasicData(name_h, i2s(value), TYPE_INT, position);
01035 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

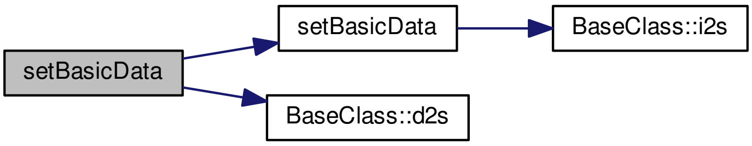


4.27.3.74 void setBasicData (const string & *name_h*, double *value*, int *position* = 0)

Definition at line 1037 of file ModelData.cpp.

```
01037                                                                          {
01038   setBasicData(name_h, d2s(value), TYPE_DOUBLE, position);
01039 }
```

Here is the call graph for this function:

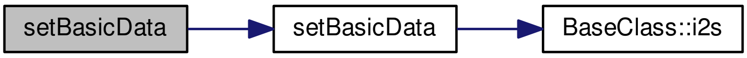


4.27.3.75 void setBasicData (const string & *name_h*, string *value*, int *position* = 0)

Definition at line 1041 of file ModelData.cpp.

```
01041                                                                          {
01042   setBasicData(name_h, value, TYPE_STRING, position);
01043 }
```

Here is the call graph for this function:

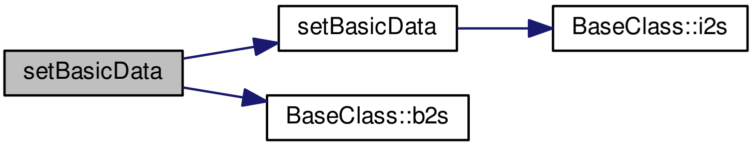


4.27.3.76 void setBasicData (const string & *name_h*, bool *value*, int *position* = 0)

Definition at line 1045 of file ModelData.cpp.

```
01045                                                                        {
01046   setBasicData(name_h, b2s(value), TYPE_BOOL, position);
01047 }
```

Here is the call graph for this function:

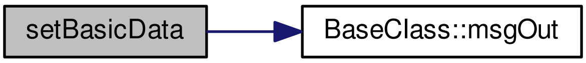


4.27.3.77 void setBasicData (const string & *name_h*, string *value*, int *type_h*, int *position*) [private]

Definition at line 1050 of file ModelData.cpp.

```
01050                                                                                {
01051   for (uint i=0; i<programSettingsVector.size();i++){
01052     if (programSettingsVector.at(i).name == name_h){
01053       int type = programSettingsVector.at(i).type;
01054       if(type != type_h){msgOut(MSG_CRITICAL_ERROR, "mismatching type in calling
      setBasicData() for "+name_h);}
01055       if(programSettingsVector.at(i).values.size() > ((uint)position)) {
01056         programSettingsVector.at(i).values.at(position)=value;
01057         return;
01058       }
01059       else {msgOut(MSG_CRITICAL_ERROR, "out-of-bound error calling setBasicData()
      for "+name_h); }
01060     }
01061   }
01062   msgOut(MSG_CRITICAL_ERROR, "Error calling setBasicData() for "+ name_h +". No
      setting option or macro data found with this name.");
01063   return;
01064 }
```

Here is the call graph for this function:



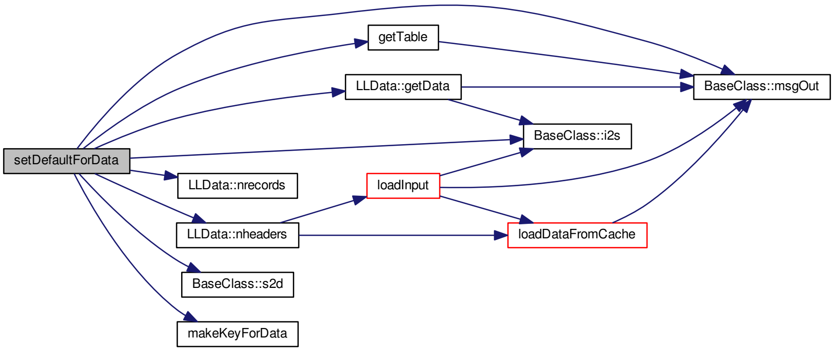
4.27.3.78 void setDefaultForData ( )

Definition at line 453 of file ModelData.cpp.

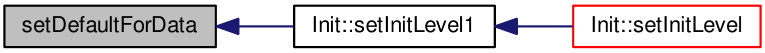
Referenced by Init::setInitLevel1().

```
00453                               {
00454   msgOut(MSG_DEBUG,"Loading forest sector data..");
00455   LLData table = getTable("forData");
00456   int nheaders = table.nheaders();
00457   for (int i=0; i< table.nrecords();i++){
00458     vector <double> values;
00459     for (int z=0;z<nheaders-4;z++){ // don't consider parName, region, forType and diamClass headers
00460       string toSearch = "value_"+i2s(z);
00461       string value = table.getData(i,toSearch);
00462       if (value != ""){
00463         values.push_back(s2d(value));
00464       }
00465     }
00466         string keys = makeKeyForData(table.getData(i,"parName"), table.
      getData(i,"region"),table.getData(i,"forType"),table.getData(i,"freeDim"));
00467     forDataMap.insert(std::pair<string, vector<double> >(keys, values));
00468   }
00469 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



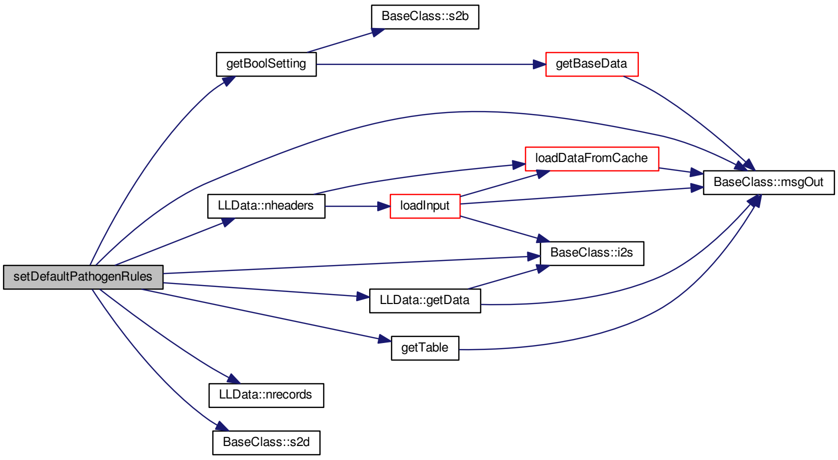
4.27.3.79 void setDefaultPathogenRules ( )

Definition at line 649 of file ModelData.cpp.

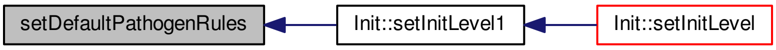
Referenced by Init::setInitLevel1().

```
00649                                        {
00650
00651   if(!getBoolSetting("usePathogenModule")) return;
00652   msgOut(MSG_DEBUG,"Loading pathogen rules..");
00653   LLData table = getTable("pathRules");
00654   int nheaders = table.nheaders();
00655   for (int i=0; i< table.nrecords();i++){
00656     pathRule PR;
00657     PR.forType = table.getData(i,"forType");
00658     PR.dClass = table.getData(i,"dClass");
00659     PR.pathId = table.getData(i,"path_name");
00660     PR.pres_min = s2d(table.getData(i,"pres_min"));
00661 
00662     vector <double> values;
00663     for (int z=0;z<nheaders-4;z++){ // don’t consider forType, dClass, path_name and pres_min  headers
00664       string toSearch = "year_"+i2s(z);
00665       string value = table.getData(i,toSearch);
00666       if (value != ""){
00667         values.push_back(s2d(value));
00668       }
00669     }
00670     PR.mortCoefficents = values;
00671 
00672     pathRules.push_back(PR);
00673   }
00674 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



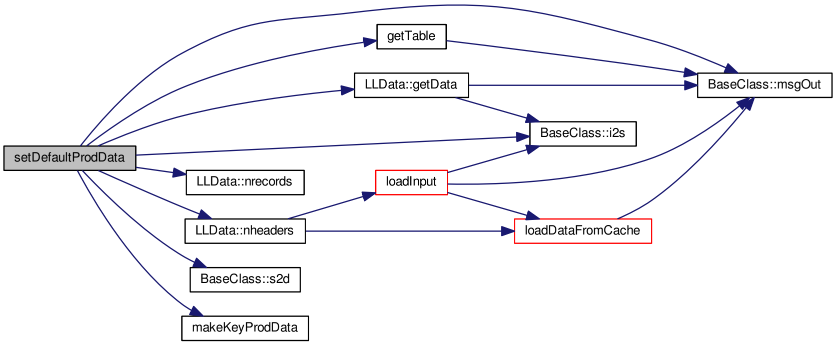
4.27.3.80 void setDefaultProdData ( )

Definition at line 503 of file ModelData.cpp.

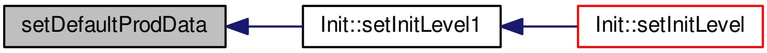
Referenced by Init::setInitLevel1().

```
00503                               {
00504 
00505   msgOut(MSG_DEBUG,"Loading products data..");
00506   LLData table = getTable("prodData");
00507   int nheaders = table.nheaders();
00508 
00509   for (int i=0; i< table.nrecords();i++){
00510 //    prodData PDATA;
00511 //    PDATA.parName = table.getData(i,"parName");
00512 //    PDATA.region = s2i(table.getData(i,"region"));
00513 //    PDATA.prod = table.getData(i,"prod");
00514 //    PDATA.freeDim = table.getData(i,"freeDim");
00515     vector <double> values;
00516     for (int z=0;z<nheaders-4;z++){ // don't consider parName, region, prod and freeDim headers
00517       string toSearch = "value_"+i2s(z);
00518       string value = table.getData(i,toSearch);
00519       if (value != ""){
00520         values.push_back(s2d(value));
00521       }
00522     }
00523 //    PDATA.values = values;
00524 //    prodDataVector.push_back(PDATA);
00525     string keys = makeKeyProdData(table.getData(i,"parName"), table.
      getData(i,"region"),table.getData(i,"prod"),table.getData(i,"freeDim"));
00526     prodDataMap.insert(std::pair<string, vector<double> >(keys, values));
00527     //giving a link to it to its own region:
00528 //    getRegion(PDATA.region)->addProdData(&PDATA);
00529   }
00530 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.3.81 void setDefaultProductResourceMatrixLink ( )

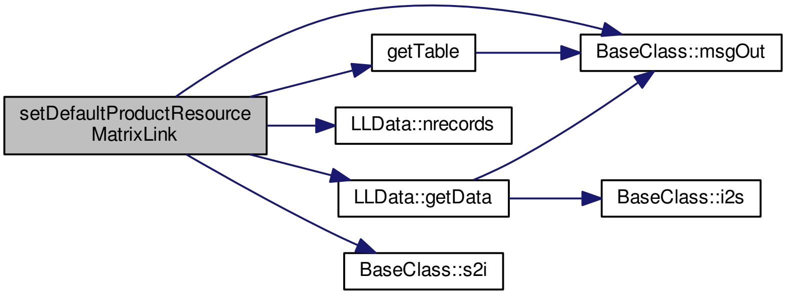
Definition at line 589 of file ModelData.cpp.

Referenced by Init::setInitLevel1().

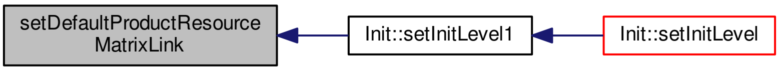
```
00589                                             {
00590   msgOut(MSG_DEBUG,"Loading forest resource to primary products io matrix..");
00591   LLData table = getTable("forToProd");
00592   for (int i=0; i< table.nrecords();i++){
00593     forToProd F2PDATA;
```

```
    F2PDATA.product = table.getData(i,"product");
    F2PDATA.forType = table.getData(i,"forType");
    F2PDATA.dClass = table.getData(i,"dClass");
    F2PDATA.maxYears = s2i(table.getData(i,"maxYears"));
    forToProdVector.push_back(F2PDATA);
  }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.3.82 void setDefaultSettings ( )

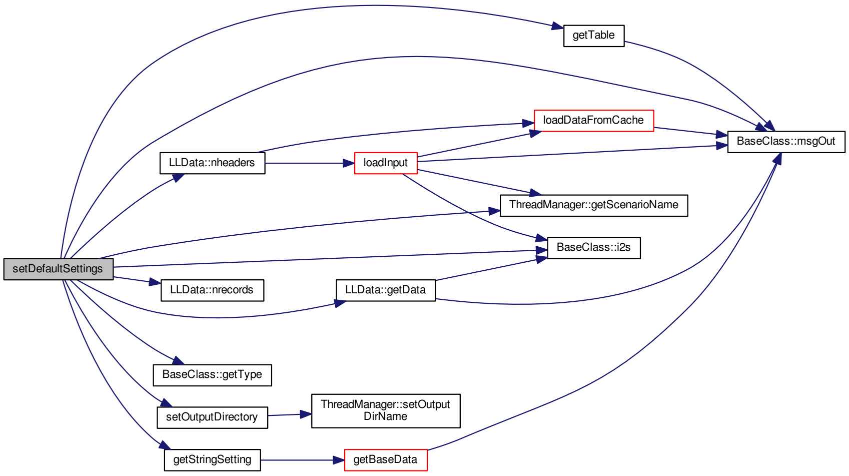
Definition at line 189 of file ModelData.cpp.

Referenced by Init::setInitLevel1().

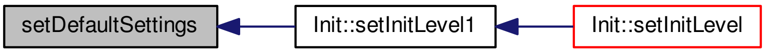
```
                                 {

  LLData table = getTable("settings");
  int nheaders = table.nheaders();
  for (int i=0; i< table.nrecords();i++){
    BasicData SETT;
    SETT.name = table.getData(i,"name");
    string type = table.getData(i,"type");
    SETT.type = getType(type);
    SETT.comment = table.getData(i,"comment");
    vector <string> values;
    for (int z=0;z<nheaders-3;z++){ // don’t consider name, type and comment headers
      string toSearch = "value_"+i2s(z);
      string value = table.getData(i,toSearch);
      if (value != ""){
        values.push_back(value);
      }
    }
    SETT.values = values;
    programSettingsVector.push_back(SETT);
```

```
00209   }
00210 
00211   msgOut(MSG_INFO,"### USING SCENARIO: "+MTHREAD->
      getScenarioName()+" ###");
00212 
00213   setOutputDirectory(getStringSetting("outputDirname").c_str());
00214 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



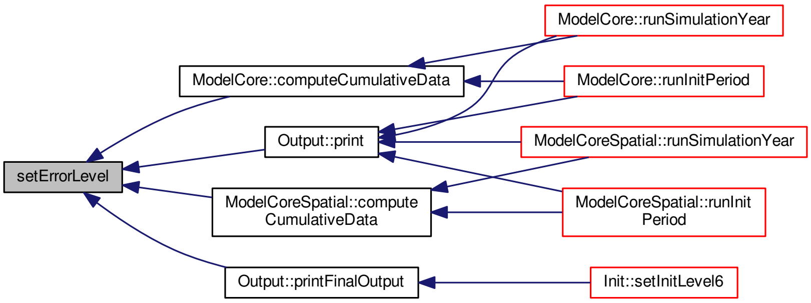
4.27.3.83 int setErrorLevel (int *errorLevel_h*) [inline]

Definition at line 141 of file ModelData.h.

Referenced by ModelCore::computeCumulativeData(), ModelCoreSpatial::computeCumulativeData(), Output::print(), and Output::printFinalOutput().

```
00141 {errorLevel=errorLevel_h;}
```

Here is the caller graph for this function:


setErrorLevel
ModelCore::computeCumulativeData
Output::print
ModelCoreSpatial::compute CumulativeData
Output::printFinalOutput
ModelCore::runSimulationYear
ModelCore::runInitPeriod
ModelCoreSpatial::runSimulationYear
ModelCoreSpatial::runInit Period
Init::setInitLevel6

4.27.3.84 void setForData (const double & *value_h*, const string & *type_h*, const int & *regId_h*, const string & *forType_h*, const string & *freeDim_h*, const int & *year* = DATA_NOW, const bool & *allowCreate* = `false`)

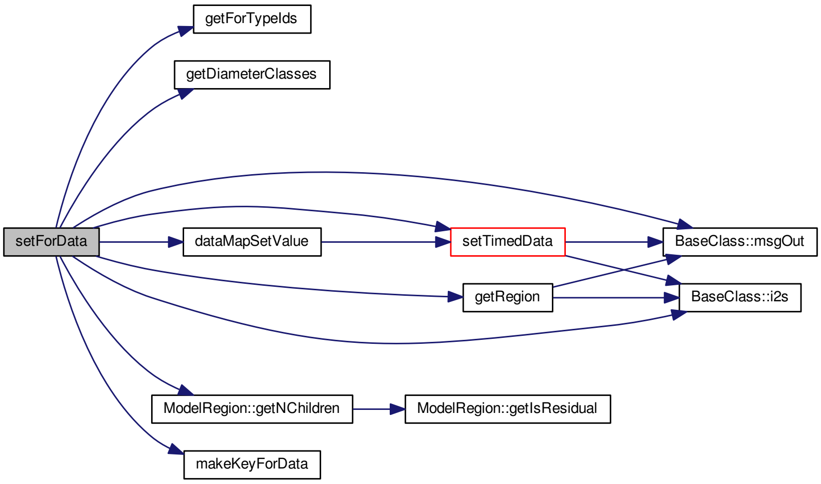
Definition at line 1302 of file ModelData.cpp.

Referenced by ModelCore::sfd(), and ModelCoreSpatial::sfd().

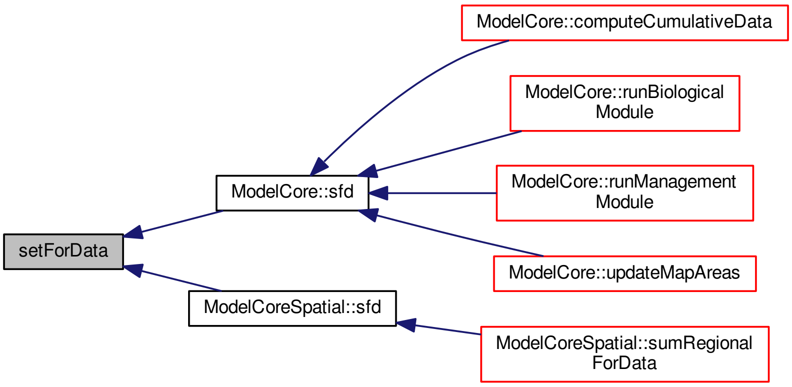
```
01302
                                                                            {
01303
01304   vector<int> regIds;
01305   vector <string> dClasses;
01306   vector <string> fTypes;
01307   string key;
01308   DataMap::const_iterator p;
01309   bool found = false;
01310   bool tempFound = false;
01311
01312   if (forType_h == FT_ALL){
01313     fTypes = getForTypeIds();
01314   } else {
01315     fTypes.push_back(forType_h);
01316   }
01317
01318     if(freeDim_h == DIAM_ALL){
01319     dClasses = diamClasses;
01320     } else if (freeDim_h == DIAM_PROD){
01321     dClasses = getDiameterClasses(true);
01322     } else if (freeDim_h == DIAM_FIRST){
01323     dClasses.push_back(diamClasses.at(0));
01324   } else  {
01325         dClasses.push_back(freeDim_h);
01326   }
01327
01328   // Make sure to set the new value to all l2 regions if requested for a reg1 level
01329   if(getRegion(regId_h)->getRegLevel()==2){
01330     regIds.push_back(regId_h);
01331   } else if (getRegion(regId_h)->getRegLevel()==1) {
01332     for(uint i=0;i<getRegion(regId_h)->getNChildren();i++){
01333       regIds.push_back(getRegion(regId_h)->getChildren()[i]->getRegId());
01334     }
01335   } else {
01336     msgOut(MSG_CRITICAL_ERROR, "Error in setProdData(). Setting a value for the
       whole World is not supported.");
01337   }
01338   int regIdsS = regIds.size();
01339
01340   for(uint r=0;r< regIds.size();r++){
01341     for(uint i=0;i<dClasses.size();i++){
01342       for (uint y=0;y<fTypes.size();y++){
01343         key = makeKeyForData(type_h,i2s(regIds[r]),fTypes[y],dClasses[i]);
01344         tempFound = dataMapSetValue(forDataMap,key,value_h, year,true);
01345         if(tempFound) found = true;
01346       }
01347     }
```

```
  }

  if(!found){
    if(!allowCreate){
            msgOut(MSG_CRITICAL_ERROR, "Error in setForData: no combination found for "+type_h+", "+i2s(regId_h)+", "+forType_h+", "+i2s(year)+", "+freeDim_h+". You can allow new variables to be created using the \"allowCreate\" flag.");
    } else {
      for(uint r=0;r< regIds.size();r++){
        for(uint i=0;i<dClasses.size();i++){
          for (uint y=0;y<fTypes.size();y++){
            key = makeKeyForData(type_h,i2s(regIds[r]),fTypes[y],dClasses[i]);
            vector <double> values;
            setTimedData(value_h,values,year,MSG_NO_MSG);
            forDataMap.insert(DataPair(key,values));
          }
        }
      }
    }
  }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:


ModelCore::computeCumulativeData
ModelCore::runBiological Module
ModelCore::sfd
ModelCore::runManagement Module
setForData
ModelCore::updateMapAreas
ModelCoreSpatial::sfd
ModelCoreSpatial::sumRegional ForData

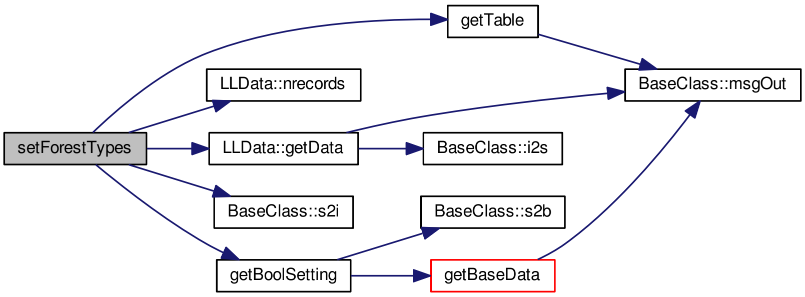
4.27.3.85 void setForestTypes ( )

Definition at line 619 of file ModelData.cpp.

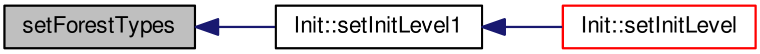
Referenced by Init::setInitLevel1().

```
00619                          {
00620   LLData table = getTable("forTypes");
00621   for (int i=0; i< table.nrecords();i++){
00622     forType FTYPE;
00623     FTYPE.forTypeId = table.getData(i,"forTypeId");
00624     FTYPE.forLabel = table.getData(i,"forLabel");
00625     FTYPE.memType = s2i(table.getData(i,"memType"));
00626     FTYPE.forLayer = table.getData(i,"forLayer");
00627     FTYPE.ereditatedFrom = table.getData(i,"ereditatedFrom");
00628     if(FTYPE.memType == 3 && !getBoolSetting("useSpExplicitForestTypes")) continue;
00629     forTypes.push_back(FTYPE);
00630   }
00631 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



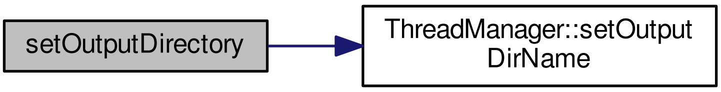
4.27.3.86 void setOutputDirectory (const char ∗ *output_dirname_h*)

Definition at line 943 of file ModelData.cpp.

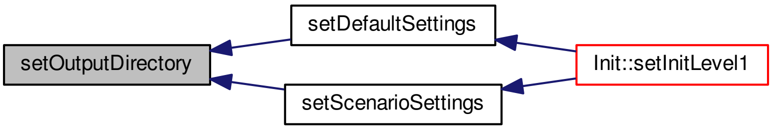
Referenced by setDefaultSettings(), and setScenarioSettings().

```
00943                                                         {
00944 
00945   if (strlen(output_dirname_h)==0){
00946     outputDirname=baseDirectory+"output/";
00947   }
00948   else {
00949     outputDirname=output_dirname_h;
00950   }
00951   MTHREAD->setOutputDirName(outputDirname); //for the GUI
00952 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.3.87 void setProdData (const double & *value_h,* const string & *type_h,* const int & *regId_h,* const string & *prodId_h,* const int & *year* = DATA_NOW*,* const bool & *allowCreate* = `false`*,* const string & *freeDim_h* = " ")

Basic function to set products-related data. It can change an existing value or extend in time a serie, but it requires the keys (par. name/regId/prodId/freedim) to be already present in the data. New value to change with/add It addmits the following "filters": Name of the specific parameter requested Set a specific level 2 region, or all its childred l2 region if a reg1 level is specified. Product. It accept three keywords, for changing/inserting the new value to all products, primary products or secondary products, namelly PROD_ALL, PROD_PRI, PROD_SEC. Unless specified, set the value of the current year. If array is smaller (e.g. because it is time-independent) fill all the values till the requested one. If true, allow creation of new data if not found. Default false (rise an error) If specified, look exactly for it, otherwise simply doesn't filter for it.

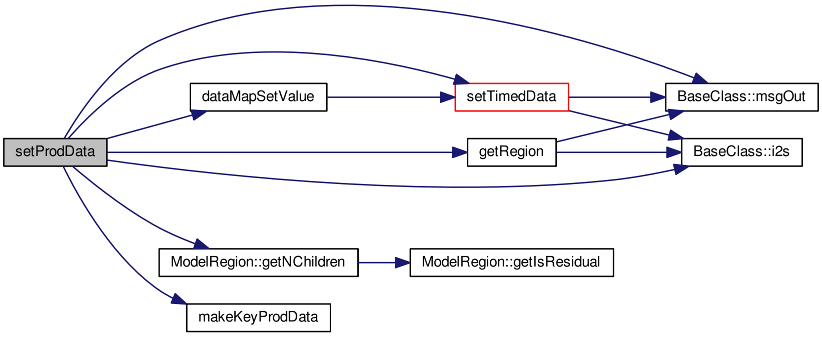
Definition at line 1242 of file ModelData.cpp.

Referenced by ModelCore::spd(), and ModelCoreSpatial::spd().

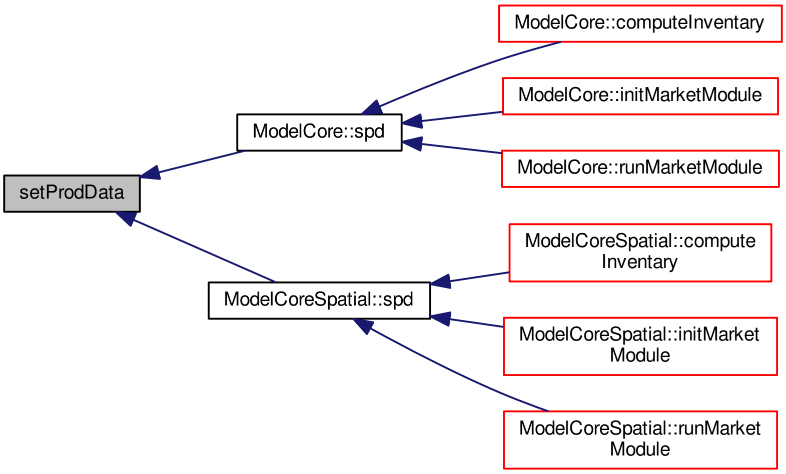
```
01242
                                                                                   {
01243
01244   vector<int> regIds;
01245   string key;
01246   DataMap::const_iterator p;
01247   vector <string> products;
01248
01249   if(prodId_h == PROD_PRI){
01250     products = priProducts;
01251   } else if (prodId_h == PROD_SEC){
01252     products = secProducts;
01253   } else if (prodId_h == PROD_ALL){
01254     products = allProducts;
01255   } else  {
01256     products.push_back(prodId_h);
01257   }
01258
01259   // Make sure to set the new value to all l2 regions if requested fora reg1 level
```

```
  if(getRegion(regId_h)->getRegLevel()==2){
    regIds.push_back(regId_h);
  } else if (getRegion(regId_h)->getRegLevel()==1) {
    for(uint i=0;i<getRegion(regId_h)->getNChildren();i++){
      regIds.push_back(getRegion(regId_h)->getChildren()[i]->getRegId());
    }
  } else {
    msgOut(MSG_CRITICAL_ERROR, "Error in setProdData(). Setting a value for the
 whole World is not supported.");
  }

  bool found = false;
  bool tempFound = false;

  for(uint r=0;r< regIds.size();r++){
    for(uint i=0;i<products.size();i++){
      key = makeKeyProdData(type_h,i2s(regIds[r]),products[i],freeDim_h);
      tempFound = dataMapSetValue(prodDataMap,key,value_h, year,true);
      if(tempFound) found = true;
    }
  }

  if(!found){
    if(!allowCreate){
      msgOut(MSG_CRITICAL_ERROR, "Error in setProdData: no combination found for "+
type_h+", "+i2s(regId_h)+", "+prodId_h+", "+i2s(year)+", "+freeDim_h+". You can allow new variables to
 be created using the \"allowCreate\" flag.");
    } else {
      for(uint r=0;r< regIds.size();r++){
        for(uint i=0;i<products.size();i++){
          key = makeKeyProdData(type_h,i2s(regIds[r]),products[i],freeDim_h);
          vector <double> values;
          setTimedData(value_h,values,year,MSG_NO_MSG);
          prodDataMap.insert(DataPair(key,values));
        }
      }
    }
  }

}
```

Here is the call graph for this function:



Here is the caller graph for this function:


ModelCore::computeInventary
ModelCore::initMarketModule
ModelCore::spd
ModelCore::runMarketModule
setProdData
ModelCoreSpatial::compute Inventary
ModelCoreSpatial::spd
ModelCoreSpatial::initMarket Module
ModelCoreSpatial::runMarket Module

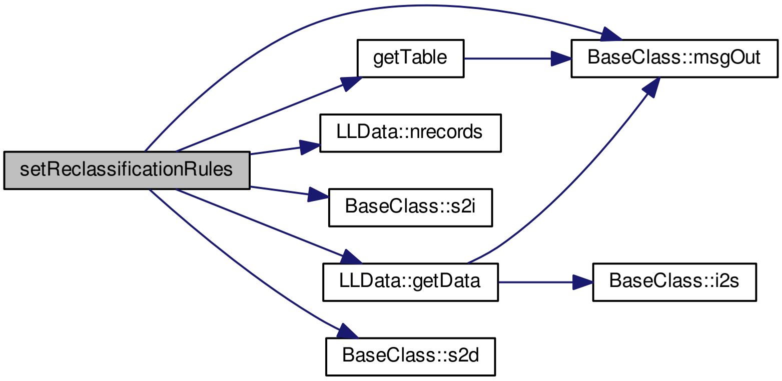
4.27.3.88 void setReclassificationRules ( )

Definition at line 634 of file ModelData.cpp.

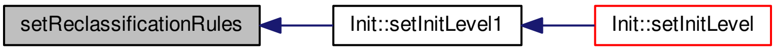
Referenced by Init::setInitLevel1().

```
00634                                     {
00635
00636   msgOut(MSG_DEBUG,"Loading (but not yet applying) reclassification rules..");
00637   LLData table = getTable("reclRules");
00638   for (int i=0; i< table.nrecords();i++){
00639     reclRule RL;
00640     RL.regId = s2i(table.getData(i,"regID"));
00641     RL.forTypeIn = table.getData(i,"forTypeIn");
00642     RL.forTypeOut = table.getData(i,"forTypeOut");
00643     RL.coeff = s2d(table.getData(i,"coeff"));
00644     reclRules.push_back(RL);
00645   }
00646 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.3.89 void setScenarioData ( )

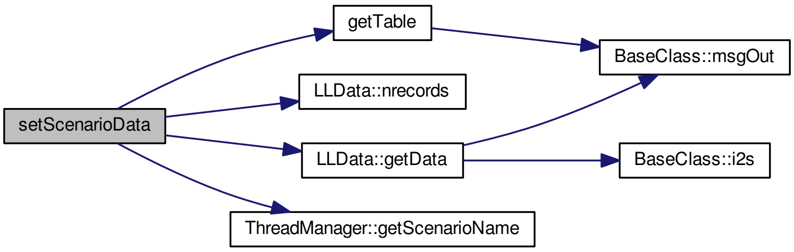
Set the infos about this scenario (long description and overriding tables)

Definition at line 168 of file ModelData.cpp.

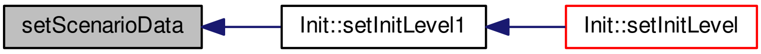
Referenced by Init::setInitLevel1().

```
00168                            {
00169   LLData table = getTable("scenarios");
00170   for(int i=0;i<table.nrecords();i++){
00171     string recordScenarioName = table.getData(i,"id");
00172     if (recordScenarioName == MTHREAD->getScenarioName()){
00173       scenario.id             = recordScenarioName;
00174       scenario.shortDesc      = table.getData(i,"shortDesc");
00175       scenario.longDesc       = table.getData(i,"longDesc");
00176       scenario.settingTable   = table.getData(i,"settingTable");
00177       scenario.forDataTable   = table.getData(i,"forDataTable");
00178       scenario.prodDataTable  = table.getData(i,"prodDataTable");
00179       scenario.forToProdTable = table.getData(i,"forToProdTable");
00180       scenario.pathTable      = table.getData(i,"pathTable");
00181       return;
00182     }
00183   }
00184
00185
00186 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



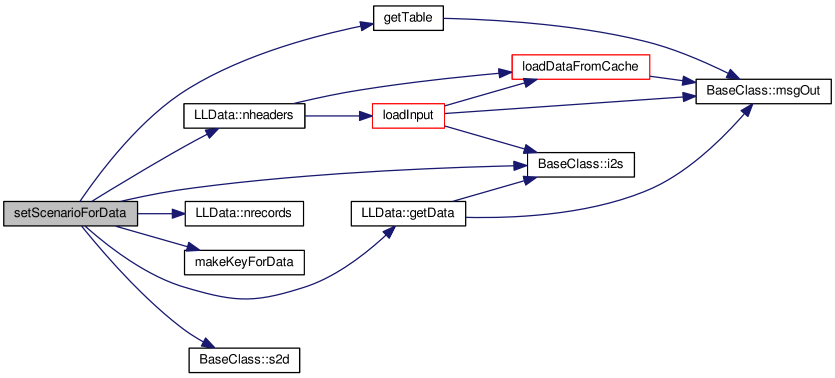
4.27.3.90 void setScenarioForData ( )

Definition at line 472 of file ModelData.cpp.

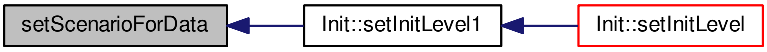
Referenced by Init::setInitLevel1().

```
00472                               {
00473 
00474   if(scenario.forDataTable==""){return;}
00475   LLData table = getTable(scenario.forDataTable,
      MSG_CRITICAL_ERROR);
00476 
00477   int nheaders = table.nheaders();
00478   for(int i=0; i< table.nrecords(); i++){
00479     bool found = false;
00480         string key = makeKeyForData(table.getData(i,"parName"),table.
      getData(i,"region"),table.getData(i,"forType"),table.getData(i,"freeDim"));
00481     vector <double> values;
00482     for (int z=0;z<nheaders-4;z++){ // don't consider parName, region, forType and diamClass headers
00483       string toSearch = "value_"+i2s(z);
00484       string value = table.getData(i,toSearch);
00485       if (value != ""){
00486         values.push_back(s2d(value));
00487       }
00488     }
00489     map <string, vector < double > >::iterator p;
00490     p=forDataMap.find(key);
00491     if(p != forDataMap.end()) {
00492       // updating an existing record
00493       p->second = values;
00494     }
00495     else {
00496       // new one, adding it
00497       forDataMap.insert(std::pair<string, vector<double> >(key, values));
00498     }
00499   }
00500 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



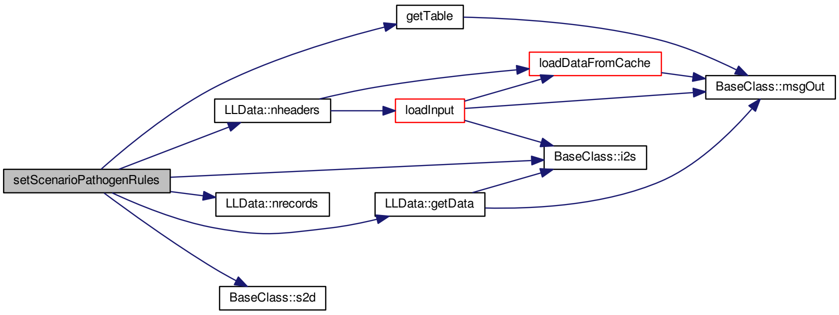
4.27.3.91 void setScenarioPathogenRules ( )

Definition at line 677 of file ModelData.cpp.

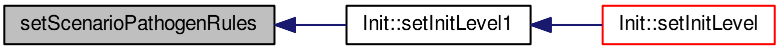
Referenced by Init::setInitLevel1().

```
00677                                  {
00678
00679   if(scenario.pathTable==""){return;}
00680   LLData table = getTable(scenario.pathTable,
      MSG_CRITICAL_ERROR); //this scenario could not have an associated setting sheet
00681
00682   int nheaders = table.nheaders();
00683   for (int i=0; i< table.nrecords();i++){
00684     pathRule PR;
00685     PR.forType = table.getData(i,"forType");
00686     PR.dClass = table.getData(i,"dClass");
00687     PR.pathId = table.getData(i,"path_name");
00688     PR.pres_min = s2d(table.getData(i,"pres_min"));
00689
00690     vector <double> values;
00691     for (int z=0;z<nheaders-4;z++){ // don’t consider forType, dClass, path_name and pres_min  headers
00692       string toSearch = "year_"+i2s(z);
00693       string value = table.getData(i,toSearch);
00694       if (value != ""){
00695         values.push_back(s2d(value));
00696       }
00697     }
00698     PR.mortCoefficents = values;
00699
00700     bool found = false;
00701     for(uint i=0;i<pathRules.size();i++){
00702       if(    pathRules[i].forType == PR.forType
00703           && pathRules[i].dClass == PR.dClass
00704           && pathRules[i].pathId == PR.pathId
00705          ){
00706         pathRules[i].pres_min = PR.pres_min;
00707         pathRules[i].mortCoefficents = PR.mortCoefficents;
00708         found = true;
00709         break;
00710       }
00711     }
00712     if(!found){
00713       pathRules.push_back(PR);
00714     }
00715   } // end for each table record
00716 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



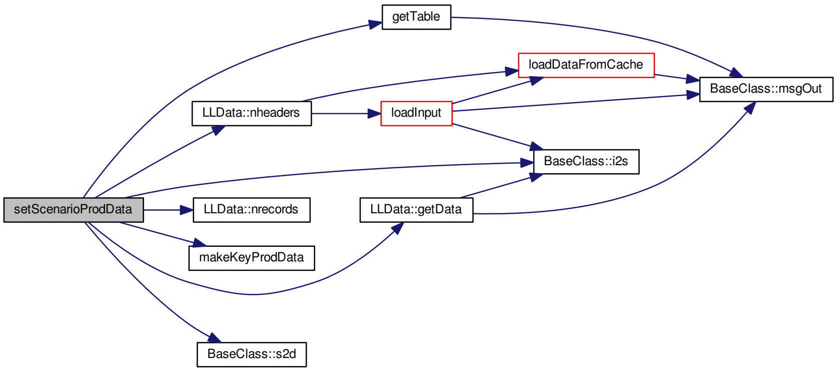
4.27.3.92 void setScenarioProdData ( )

Definition at line 533 of file ModelData.cpp.

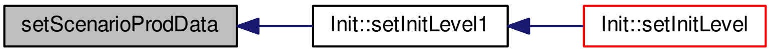
Referenced by Init::setInitLevel1().

```
00533                                 {
00534 
00535   if(scenario.prodDataTable==""){return;}
00536   LLData table = getTable(scenario.prodDataTable,
      MSG_CRITICAL_ERROR); //this scenario could not have an associated setting sheet
00537 
00538   int nheaders = table.nheaders();
00539   for(int i=0; i< table.nrecords(); i++){
00540     //prodData PDATA;
00541     bool found = false;
00542     string key = makeKeyProdData(table.getData(i,"parName"),table.
      getData(i,"region"),table.getData(i,"prod"),table.getData(i,"freeDim"));
00543 
00544     //PDATA.parName = table.getData(i,"parName");
00545     //PDATA.region = s2i(table.getData(i,"region"));
00546     //PDATA.prod = table.getData(i,"prod");
00547     //PDATA.freeDim = table.getData(i,"freeDim");
00548     vector <double> values;
00549     for (int z=0;z<nheaders-4;z++){// don’t consider parName, region, prod and freeDim headers
00550       string toSearch = "value_"+i2s(z);
00551       string value = table.getData(i,toSearch);
00552       if (value != ""){
00553         values.push_back(s2d(value));
00554       }
00555     }
00556     //PDATA.values = values;
00557     //for(uint i=0;i<prodDataVector.size();i++){
00558     //  if(prodDataVector[i].parName == PDATA.parName
00559     //     && prodDataVector[i].region == PDATA.region
00560     //     && prodDataVector[i].prod == PDATA.prod
00561     //     && prodDataVector[i].freeDim == PDATA.freeDim){
00562     //    // existing prodData..
00563     //    prodDataVector[i].values = PDATA.values;
00564     //    found = true;
00565     //    break;
00566     //  }
00567     //}
00568     //if(!found){
00569     //  // new one, adding it
00570     //  prodDataVector.push_back(PDATA);
00571     //  //giving a link to it to its own region:
00572     //  getRegion(PDATA.region)->addProdData(&PDATA);
00573     //}
00574 
00575     map <string, vector < double > >::iterator p;
00576     p=prodDataMap.find(key);
00577     if(p != prodDataMap.end()) {
00578       // updating an existing record
00579       p->second = values;
00580     }
00581     else {
00582       // new one, adding it
00583       prodDataMap.insert(std::pair<string, vector<double> >(key, values));
00584     }
00585   }
00586 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



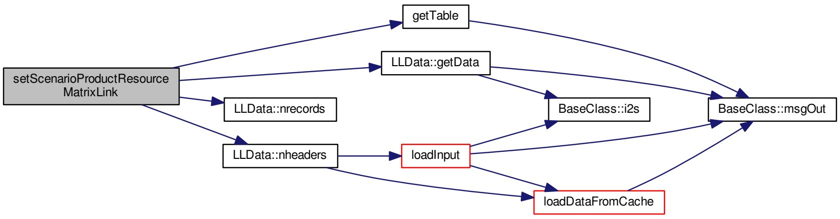
4.27.3.93 void setScenarioProductResourceMatrixLink ( )

Definition at line 603 of file ModelData.cpp.

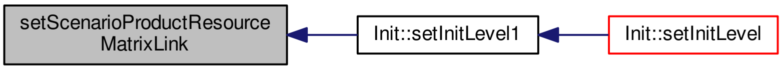
Referenced by Init::setInitLevel1().

```
00603                                              {
00604   if(scenario.forToProdTable==""){return;}
00605   LLData table = getTable(scenario.forToProdTable,
      MSG_CRITICAL_ERROR); //this scenario could not have an associated setting sheet
00606 
00607   int nheaders = table.nheaders();
00608   forToProdVector.clear();
00609   for (int i=0; i< table.nrecords();i++){
00610     forToProd F2PDATA;
00611     F2PDATA.product = table.getData(i,"product");
00612     F2PDATA.forType = table.getData(i,"forType");
00613     F2PDATA.dClass = table.getData(i,"dClass");
00614     forToProdVector.push_back(F2PDATA);
00615   }
00616 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



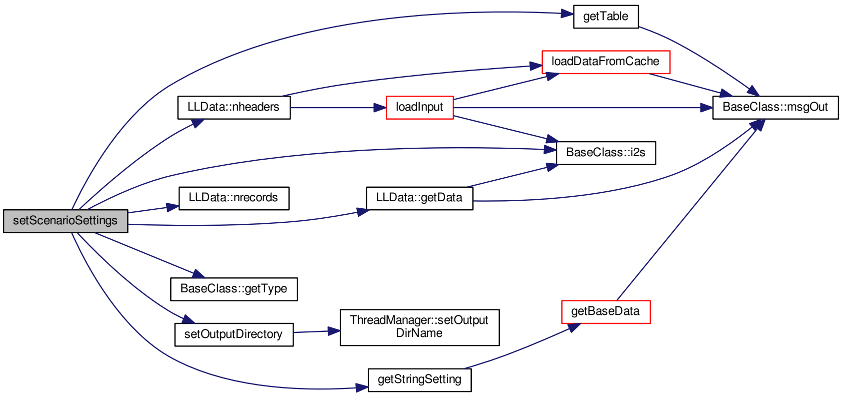
4.27.3.94 void setScenarioSettings ( )

Definition at line 217 of file ModelData.cpp.

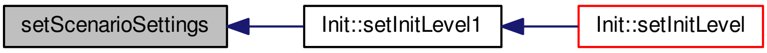
Referenced by Init::setInitLevel1().

```
00217                               {
00218 
00219   if(scenario.settingTable=="") {return;}
00220   LLData table = getTable(scenario.settingTable,
      MSG_CRITICAL_ERROR); //this scenario could not have an associated setting sheet
00221 
00222   int nheaders = table.nheaders();
00223   for(int i=0; i< table.nrecords(); i++){
00224     BasicData SETT;
00225     string name = table.getData(i,"name");
00226     string stype = table.getData(i,"type");
00227     int type = getType(stype);
00228     string comment = table.getData(i,"comment");
00229     vector <string> values;
00230     for (int z=0;z<nheaders-3;z++){ // don’t consider name, type and comment headers
00231       string toSearch = "value_"+i2s(z);
00232       string value = table.getData(i,toSearch);
00233       if (value != ""){
00234         values.push_back(value);
00235       }
00236     }
00237 
00238     for(uint i=0;i<programSettingsVector.size();i++){
00239       if(programSettingsVector[i].name == name){
00240         programSettingsVector[i].values = values;
00241         programSettingsVector[i].type = type;
00242         programSettingsVector[i].comment = comment;
00243         break;
00244       }
00245     }
00246 
00247   }
00248 
00249   setOutputDirectory(getStringSetting("outputDirname").c_str());
00250 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.3.95 void setSpace ( )

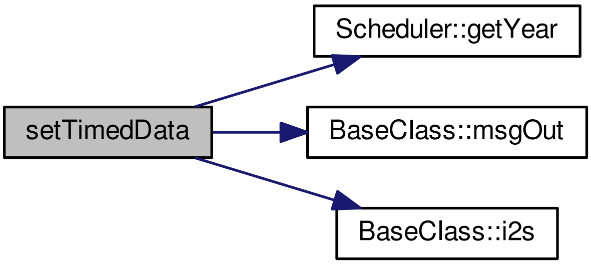
4.27.3.96 void setTimedData (const double & *value_h,* vector< double > & *dated_vector,* const int & *year_h,* const int & *MSG_LEVEL* = MSG_WARNING)

Definition at line 1391 of file ModelData.cpp.

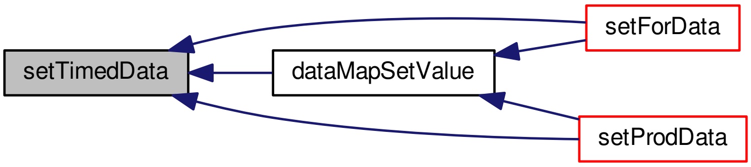
Referenced by dataMapSetValue(), setForData(), and setProdData().

```
01391
               {
01392
01393   int position;
01394   if(year_h==DATA_NOW){
01395     position = MTHREAD->SCD->getYear()-cached_initialYear;
01396   } else {
01397     position = year_h-cached_initialYear;
01398   }
01399
01400   int originalVectorSize = dated_vector.size();
01401   if(dated_vector.size() > position) {
01402     dated_vector[position]=value_h;
01403   } else {
01404     // extending the vector and filling it with the incoming value, but issuing a warning if done for more
       than one year
01405
01406     for(uint i=0;i<position-originalVectorSize+1;i++){
01407       dated_vector.push_back(value_h);
01408     }
01409     if(position-originalVectorSize > 0 ){
01410       msgOut(MSG_LEVEL, "setTimedData: a dated vector has been filled several years ("+
      i2s(1+position-originalVectorSize)+") with incoming values to reach desidered position in time.");
01411     }
01412   }
01413 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



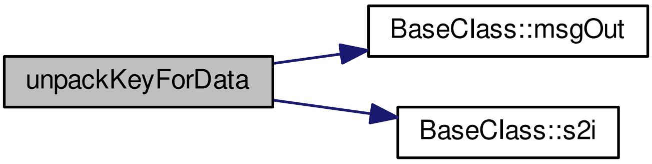
4.27.3.97 void unpackKeyForData (const string & *key*, string & *parName*, int & *regId*, string & *forType*, string & *diamClass*) const

Definition at line 1778 of file ModelData.cpp.

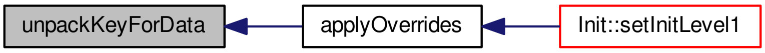
Referenced by applyOverrides().

```
01778
              {
01779   int parNameDelimiter = key.find("#",0);
01780   int regIdDelimiter = key.find("#",parNameDelimiter+1);
01781   int forTypeDelimiter = key.find("#",regIdDelimiter+1);
01782   int diamClassDelimiter = key.find("#",forTypeDelimiter+1);
01783   if (diamClassDelimiter == string::npos){
01784     msgOut(MSG_CRITICAL_ERROR, "Error in unpacking a key in the map of production
       data.");
01785   }
01786   parName.assign(key,0,parNameDelimiter);
01787   string regIdString="";
01788   regIdString.assign(key,parNameDelimiter+1,regIdDelimiter-parNameDelimiter-1);
01789   regId = s2i(regIdString);
01790   forType.assign(key,regIdDelimiter+1,forTypeDelimiter-regIdDelimiter-1);
01791   diamClass.assign(key,forTypeDelimiter+1,diamClassDelimiter-forTypeDelimiter-1);
01792
01793 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



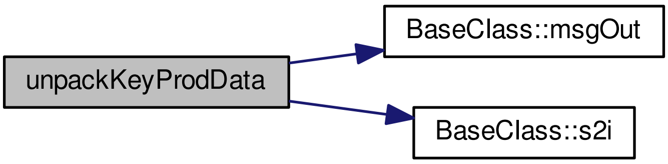
4.27.3.98 void unpackKeyProdData (const string & *key,* string & *parName,* int & *regId,* string & *prod,* string & *freeDim*) const

Definition at line 1759 of file ModelData.cpp.

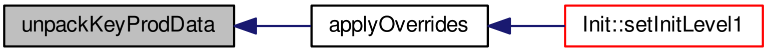
Referenced by applyOverrides().

```
01759
          {
01760
01761   int parNameDelimiter = key.find("#",0);
01762   int regIdDelimiter = key.find("#",parNameDelimiter+1);
01763   int prodDelimiter = key.find("#",regIdDelimiter+1);
01764   int freeDimDelimiter = key.find("#",prodDelimiter+1);
01765   if (freeDimDelimiter == string::npos){
01766     msgOut(MSG_CRITICAL_ERROR, "Error in unpacking a key in the map of production
       data.");
01767   }
01768   parName.assign(key,0,parNameDelimiter);
01769   string regIdString="";
01770   regIdString.assign(key,parNameDelimiter+1,regIdDelimiter-parNameDelimiter-1);
01771   regId = s2i(regIdString);
01772   prod.assign(key,regIdDelimiter+1,prodDelimiter-regIdDelimiter-1);
01773   freeDim.assign(key,prodDelimiter+1,freeDimDelimiter-prodDelimiter-1);
01774
01775 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.27.4 Friends And Related Function Documentation

4.27.4.1 void **Output::printForestData** (bool *finalFlush* = `false`) `[friend]`

4.27.4.2 void **Output::printProductData** (bool *finalFlush* = `false`) `[friend]`

4.27.5 Member Data Documentation

4.27.5.1 vector<string> allProducts `[private]`

Definition at line 230 of file ModelData.h.

Referenced by cacheSettings(), getProdData(), and setProdData().

4.27.5.2 string baseDirectory `[private]`

Definition at line 208 of file ModelData.h.

Referenced by getFilenameByType(), and setOutputDirectory().

4.27.5.3 int cached_initialYear `[private]`

Definition at line 227 of file ModelData.h.

Referenced by cacheSettings(), getTimedData(), and setTimedData().

4.27.5.4 map<iisskey, double > deathTimberInventory `[private]`

Map that register the death of biomass still usable as timber by year, l2_region, forest type and diameter class [Mm^3 wood].

Definition at line 223 of file ModelData.h.

Referenced by getAvailableDeathTimber().

4.27.5.5 vector<string> diamClasses `[private]`

Diameter classes.

Definition at line 226 of file ModelData.h.

Referenced by cacheSettings(), getAvailableAliveTimber(), getAvailableDeathTimber(), getDiameterClasses(), getForData(), and setForData().

4.27.5.6 int errorLevel `[private]`

Definition at line 236 of file ModelData.h.

Referenced by dataMapSetValue(), getForData(), getProdData(), and ModelData().

4.27.5.7 map< string, vector< double > > forDataMap [private]

Forestry data.

Definition at line 210 of file ModelData.h.

Referenced by applyOverrides(), getForData(), setDefaultForData(), setForData(), and setScenarioForData().

4.27.5.8 vector< **forToProd** > forToProdVector [private]

Vector of coefficients from forest resources to primary products.

Definition at line 212 of file ModelData.h.

Referenced by assessProdPossibility(), getMaxYearUsableDeathTimber(), setDefaultProductResourceMatrixLink(), and setScenarioProductResourceMatrixLink().

4.27.5.9 vector< **forType** > forTypes [private]

Vector of forest types.

Definition at line 219 of file ModelData.h.

Referenced by getForType(), getForTypeChilds(), getForTypeIds(), getForTypeParentId(), getForTypeParents(), getNForTypesChilds(), and setForestTypes().

4.27.5.10 vector< **IFiles** > iFilesVector [private]

List of all input files. Simple (struct)

Definition at line 214 of file ModelData.h.

Referenced by getFilenameByType().

4.27.5.11 string inputFilename [private]

Definition at line 206 of file ModelData.h.

4.27.5.12 vector< vector <int> > l2r [private]

Region2 ids.

Definition at line 222 of file ModelData.h.

4.27.5.13 vector< **LLData** > LLDataVector [private]

Vector of Low Level Data.

Definition at line 216 of file ModelData.h.

Referenced by getTable(), loadDataFromCache(), and loadInput().

4.27.5.14 **InputNode** mainDocument `[private]`

For each agricultural soil type (as defined in the setting "agrLandTypes") this list define the objects that can be placed on that soil type.

the main input document, loaded in memory at unzipping stage

Definition at line 235 of file ModelData.h.

Referenced by loadInput().

4.27.5.15 string outputDirname `[private]`

Definition at line 207 of file ModelData.h.

Referenced by setOutputDirectory().

4.27.5.16 vector< **pathRule** > pathRules `[private]`

Vector of pathogen rules.

Definition at line 221 of file ModelData.h.

Referenced by getPathMortalityRule(), setDefaultPathogenRules(), and setScenarioPathogenRules().

4.27.5.17 vector< string > priProducts `[private]`

Definition at line 228 of file ModelData.h.

Referenced by cacheSettings(), getAllocableProductIdsFromDeathTimber(), getProdData(), and setProdData().

4.27.5.18 map< string, vector< double > > prodDataMap `[private]`

Product data.

Definition at line 211 of file ModelData.h.

Referenced by applyOverrides(), getProdData(), setDefaultProdData(), setProdData(), and setScenarioProdData().

4.27.5.19 vector< **BasicData** > programSettingsVector `[private]`

Setting data. Simple (struct)

Definition at line 215 of file ModelData.h.

Referenced by addSetting(), applyDebugMode(), getBaseData(), getVectorBaseData(), setBasicData(), set←DefaultSettings(), and setScenarioSettings().

4.27.5.20 vector< **reclRule** > reclRules `[private]`

Vector of reclassification rules.

Definition at line 220 of file ModelData.h.

Referenced by applyOverrides(), and setReclassificationRules().

4.27.5.21 vector< **ModelRegion** > regionsVector `[private]`

Vector of modelled regions.

Definition at line 217 of file ModelData.h.

Referenced by applyDebugMode(), createRegions(), getAllRegions(), getRegion(), getRegionIds(), and region↩
Exist().

4.27.5.22 **scenarioData** scenario

Definition at line 195 of file ModelData.h.

Referenced by setScenarioData(), setScenarioForData(), setScenarioPathogenRules(), setScenarioProdData(), setScenarioProductResourceMatrixLink(), and setScenarioSettings().

4.27.5.23 vector<string> secProducts `[private]`

Definition at line 229 of file ModelData.h.

Referenced by cacheSettings(), getProdData(), and setProdData().

4.27.5.24 bool tempBool `[private]`

a temporary bool variable used for various functions

Definition at line 232 of file ModelData.h.

Referenced by dataMapGetValue(), getForData(), and getProdData().

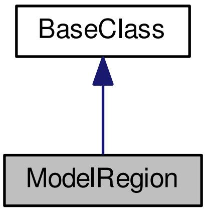
The documentation for this class was generated from the following files:

- /home/lobianco/git/ffsm_pp/src/ModelData.h
- /home/lobianco/git/ffsm_pp/src/ModelData.cpp

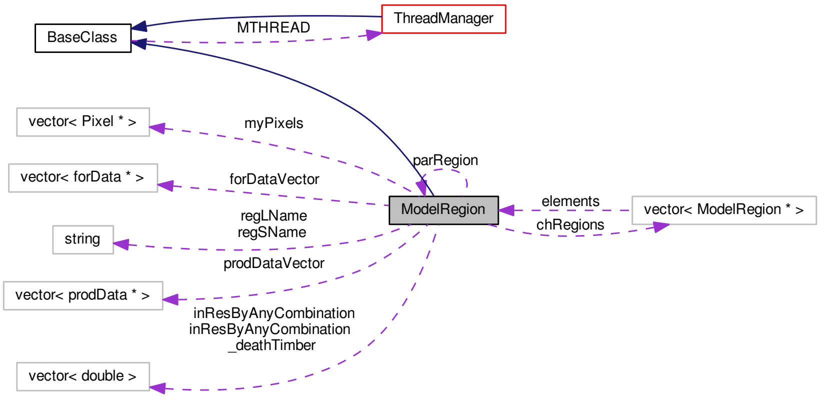
4.28 ModelRegion Class Reference

`#include <ModelRegion.h>`

Inheritance diagram for ModelRegion:



Collaboration diagram for ModelRegion:



Public Member Functions

- ModelRegion (ThreadManager ∗MTHREAD_h, int regId_h, string regSName_h, string regLName_h, int regLevel_h, int parRegId_h, bool isResidual_h)

    Constructor.
- ∼ModelRegion ()
- void setRegId (int regId_h)
- void setRegSName (string regSName_h)
- void setRegLName (string regLName_h)
- void setRegLevel (int regLevel_h)
- void setParRegId (int parRegId_h)
- void setIsResidual (bool isResidual_h)
- void setParent (ModelRegion ∗parRegion_h)
- void setChildren (vector< ModelRegion ∗ > children_h)
- void addForData (forData ∗data_h)

    Childrens are all the lvel-1 region that are parts of this region.
- void addProdData (prodData ∗data_h)
- void setMyPixels ()

    It sets a double link pixels <–> region.
- void swap (const int &swap_what)
- int getRegId () const
- string getRegSName () const
- string getRegLName () const
- int getRegLevel () const
- int getParRegId () const
- bool getIsResidual () const
- ModelRegion ∗ getParent ()
- vector< ModelRegion ∗ > getChildren (bool excludeResidual=true)

    Returns a pointer to the parent regions.
- double getVolumes ()
- vector< double > getVolumes (int fType_h)
- double getValue (string layerName, int op=OP_SUM)

    return the values of its own pixels for the specified layer. Possible operations: OP_SUM or OP_AVG
- vector< vector< double > > getVolumes (int fType_h, string dClass_h)

- double getArea (const string &fType_h, const string &dClass_h)

  Get area by ft and dc (from pixel->area matrix)
- double getArea (const string &fType_h)

  Get area by ft (from pixel->area matrix)
- double getArea (const int &ft_pos, const int &dc_pos)

  Get area by ft and dc positions (from pixel->area matrix)
- double getArea (const int &ft_pos)

  Get area by ft position (from pixel->area matrix)
- double getArea ()

  Get whole forest area (from pixel->area matrix)
- int getNChildren (bool excludeResidual=true)
- vector< Pixel * > getMyPixels ()

Public Attributes

- vector< double > inResByAnyCombination

  Vector of inventory resource for each possible combination of primary products. This store both alive timber and death one.
- vector< double > inResByAnyCombination_deathTimber

  Vector of inventory resource for each possible combination of primary products. This store only death timber.

Private Attributes

- int regId

  Regional unique ID.
- string regSName

  A short name of the region.
- string regLName

  Region long name;.
- int regLevel

  The level of the region. 1: country, 2: regions.
- int parRegId

  Id of the parent region;.
- bool isResidual

  A flag if this region should be explicitelly modelled or it is just a residual.
- ModelRegion * parRegion

  Pointer to the parent region.
- vector< ModelRegion * > chRegions

  Vector of level-1 children regions.
- vector< forData * > forDataVector

  Vector of pointers of forestry data (owned by ModelData)
- vector< prodData * > prodDataVector

  Vector of pointers of product data (owned by ModelData)
- vector< Pixel * > myPixels

  Vector of pixels for this region.

Additional Inherited Members

4.28.1 Detailed Description

Definition at line 45 of file ModelRegion.h.

4.28.2 Constructor & Destructor Documentation

4.28.2.1 ModelRegion (ThreadManager ∗ *MTHREAD_h,* int *regId_h,* string *regSName_h,* string *regLName_h,* int *regLevel_h,* int *parRegId_h,* bool *isResidual_h*)

Constructor.

The constructor of REGION instances want:

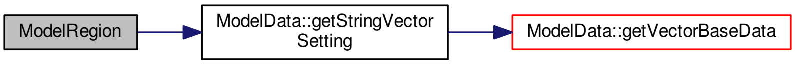
Parameters

MTHREAD_h	Pointer to the main thread manager

Definition at line 34 of file ModelRegion.cpp.

```
00034
                                                  {
00035   MTHREAD=MTHREAD_h;
00036   regId = regId_h;
00037   regSName = regSName_h;
00038   regLName = regLName_h;
00039   regLevel = regLevel_h;
00040   parRegId = parRegId_h;
00041   isResidual =  isResidual_h;
00042
00043   // Create an empty vector of inventory bounds for each possible primary products combination
00044   int nInBounds = pow(2,MTHREAD->MD->getStringVectorSetting("priProducts").
      size());
00045   //int nInBounds = MTHREAD->MD->getStringVectorSetting("priProducts").size(); // TODO todo !Important
00046   vector <double> inBounds(nInBounds,0.); // should have ceated a vector of size priProducts.size(), all
     filled with zeros
00047   inResByAnyCombination             = inBounds;
00048   inResByAnyCombination_deathTimber = inBounds;
00049 }
```

Here is the call graph for this function:


ModelRegion
ModelData::getStringVector Setting
ModelData::getVectorBaseData

4.28.2.2 ∼ModelRegion ( )

Definition at line 51 of file ModelRegion.cpp.

```
00051                          {
00052 }
```

4.28.3 Member Function Documentation

4.28.3.1 void addForData (forData ∗ *data_h*) `[inline]`

Childrens are all the lvel-1 region that are parts of this region.

Definition at line 60 of file ModelRegion.h.

```
00060 {forDataVector.push_back(data_h);};
```

4.28.3.2 void addProdData (prodData * *data_h*) [inline]

Definition at line 61 of file ModelRegion.h.

```
00061 {prodDataVector.push_back(data_h);};
```

4.28.3.3 double getArea (const string & *fType_h,* const string & *dClass_h*)

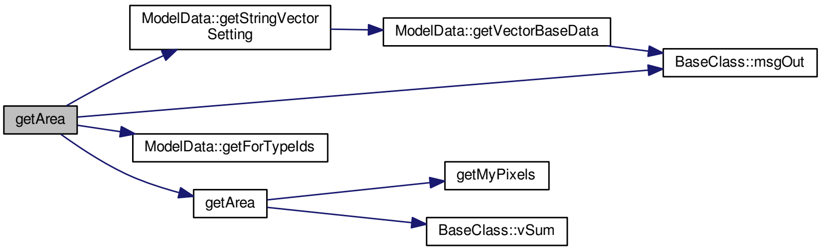
Get area by ft and dc (from pixel->area matrix)

Definition at line 106 of file ModelRegion.cpp.

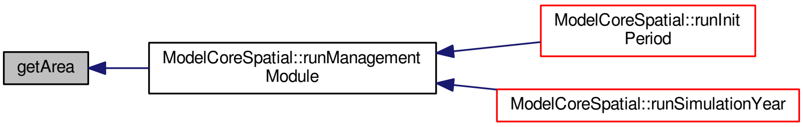
Referenced by ModelCoreSpatial::runManagementModule().

```
00106                                                                   {
00107   vector <string> dClasses = MTHREAD->MD->getStringVectorSetting("dClasses")
      ;
00108   vector <string> fTypes= MTHREAD->MD->getForTypeIds();
00109   int ft_pos = -1000;
00110   int dc_pos = -1000;
00111   for(uint j=0;j<fTypes.size();j++){
00112     if (fTypes[j] == fType_h){
00113       ft_pos = j;
00114       break;
00115     }
00116   }
00117   for(uint u=0;u<dClasses.size();u++){
00118     if (dClasses[u] == dClass_h){
00119       dc_pos = u;
00120       break;
00121     }
00122   }
00123   if(ft_pos<0) msgOut(MSG_CRITICAL_ERROR,"Forest type "+fType_h+" not found in
       getArea() function.");
00124   if(dc_pos<0) msgOut(MSG_CRITICAL_ERROR,"Diameter class"+dClass_h+" not found in
       getArea() function.");
00125
00126   return getArea(ft_pos, dc_pos);
00127 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


getArea
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

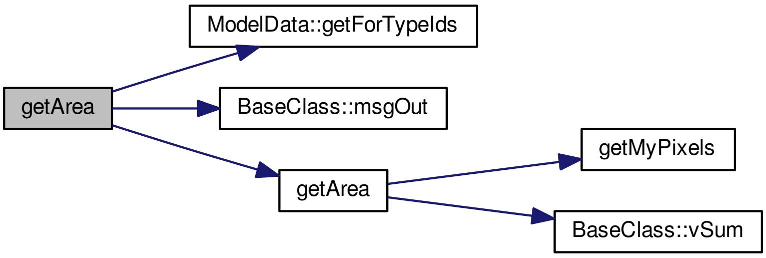
4.28.3.4 double getArea (const string & *fType_h*)

Get area by ft (from pixel->area matrix)

Definition at line 130 of file ModelRegion.cpp.

```
00130                                           {
00131   vector <string> fTypes= MTHREAD->MD->getForTypeIds();
00132   int ft_pos = -1000;
00133   for(uint j=0;j<fTypes.size();j++){
00134     if (fTypes[j] == fType_h){
00135       ft_pos = j;
00136       break;
00137     }
00138   }
00139   if(ft_pos<0) msgOut(MSG_CRITICAL_ERROR,"Forest type "+fType_h+" not found in
       getArea() function.");
00140   return getArea(ft_pos);
00141 }
```

Here is the call graph for this function:



4.28.3.5 double getArea (const int & *ft_pos*, const int & *dc_pos*)

Get area by ft and dc positions (from pixel->area matrix)

Definition at line 144 of file ModelRegion.cpp.

```
00144                                                          {
00145   double totalarea = 0.0;
00146   for(uint i=0;i<myPixels.size(); i++){
00147      totalarea += myPixels[i]->area.at(ft_pos).at(dc_pos);
00148   }
00149   return totalarea;
00150 }
```

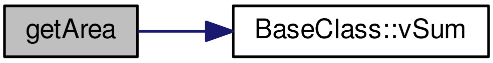
4.28.3.6 double getArea (const int & *ft_pos*)

Get area by ft position (from pixel->area matrix)

Definition at line 153 of file ModelRegion.cpp.

```
00153                                    {
00154   double totalarea = 0.0;
00155   for(uint i=0;i<myPixels.size(); i++){
00156      totalarea += vSum(myPixels[i]->area.at(ft_pos));
00157   }
00158   return totalarea;
00159 }
```

Here is the call graph for this function:



4.28.3.7 double getArea ( )

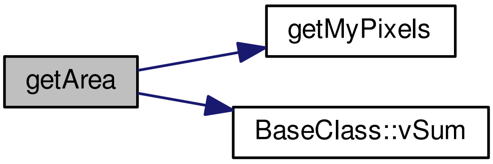
Get whole forest area (from pixel->area matrix)

Definition at line 162 of file ModelRegion.cpp.

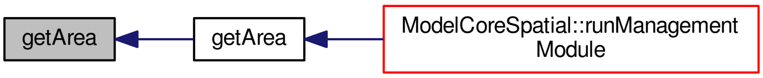
Referenced by getArea().

```
00162                          {
00163   vector<Pixel*> regPx = this->getMyPixels();
00164   double totalarea = 0.0;
00165   for(uint i=0;i<myPixels.size(); i++){
00166      totalarea += vSum(myPixels[i]->area);
00167   }
00168   return totalarea;
00169 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.28.3.8 vector< **ModelRegion** * > getChildren (bool *excludeResidual* = `true`)

Returns a pointer to the parent regions.

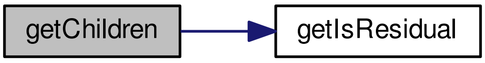
Return a vector of pointers to the direct child regions

Definition at line 55 of file ModelRegion.cpp.

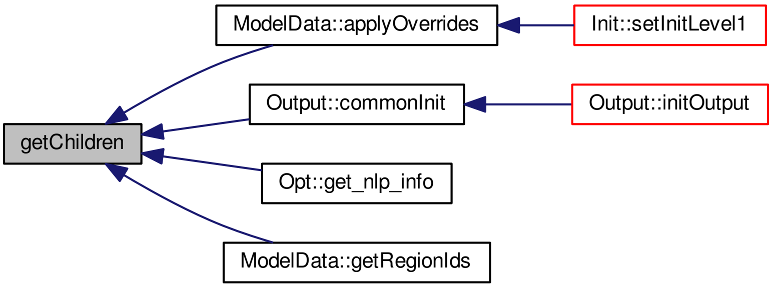
Referenced by ModelData::applyOverrides(), Output::commonInit(), Opt::get_nlp_info(), and ModelData::getRegionIds().

```
00055                                              {
00056   if(excludeResidual){
00057     vector<ModelRegion*> toReturn;
00058     for(uint i=0;i<chRegions.size();i++){
00059       if(!chRegions[i]->getIsResidual()){
00060         toReturn.push_back(chRegions[i]);
00061       }
00062     }
00063     return toReturn;
00064   }
00065   return chRegions;
00066 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



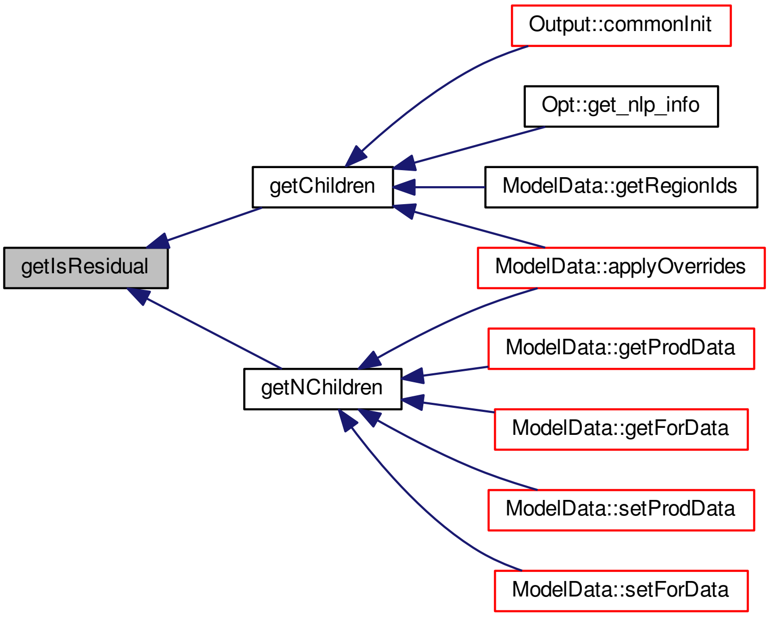
4.28.3.9 bool getIsResidual ( ) const [inline]

Definition at line 71 of file ModelRegion.h.

Referenced by getChildren(), and getNChildren().

```
00071 {return isResidual;};
```

Here is the caller graph for this function:



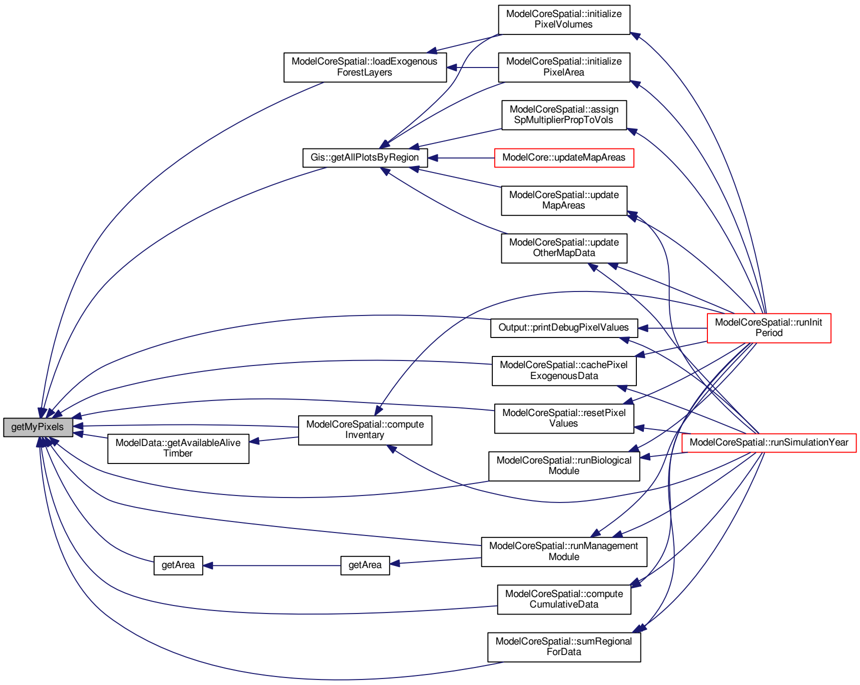
4.28.3.10 vector< Pixel∗ > getMyPixels ( ) [inline]

Definition at line 85 of file ModelRegion.h.

Referenced by ModelCoreSpatial::cachePixelExogenousData(), ModelCoreSpatial::computeCumulativeData(), ModelCoreSpatial::computeInventary(), Gis::getAllPlotsByRegion(), getArea(), ModelData::getAvailableAliveTimber(), ModelCoreSpatial::loadExogenousForestLayers(), Output::printDebugPixelValues(), ModelCoreSpatial::resetPixelValues(), ModelCoreSpatial::runBiologicalModule(), ModelCoreSpatial::runManagementModule(), and ModelCoreSpatial::sumRegionalForData().

```
00085 {return myPixels;};
```

Here is the caller graph for this function:


ModelCoreSpatial::initialize PixelVolumes
ModelCoreSpatial::loadExogenous ForestLayers
ModelCoreSpatial::initialize PixelArea
ModelCoreSpatial::assign SpMultiplierPropToVols
Gis::getAllPlotsByRegion
ModelCore::updateMapAreas
ModelCoreSpatial::update MapAreas
ModelCoreSpatial::update OtherMapData
Output::printDebugPixelValues
ModelCoreSpatial::runInit Period
ModelCoreSpatial::cachePixel ExogenousData
getMyPixels
ModelCoreSpatial::compute Inventary
ModelCoreSpatial::resetPixel Values
ModelData::getAvailableAlive Timber
ModelCoreSpatial::runSimulationYear
ModelCoreSpatial::runBiological Module
ModelCoreSpatial::runManagement Module
getArea
getArea
ModelCoreSpatial::compute CumulativeData
ModelCoreSpatial::sumRegional ForData

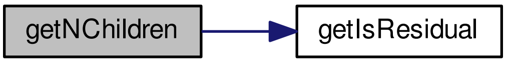
4.28.3.11 int getNChildren (bool *excludeResidual* = true)

Definition at line 69 of file ModelRegion.cpp.

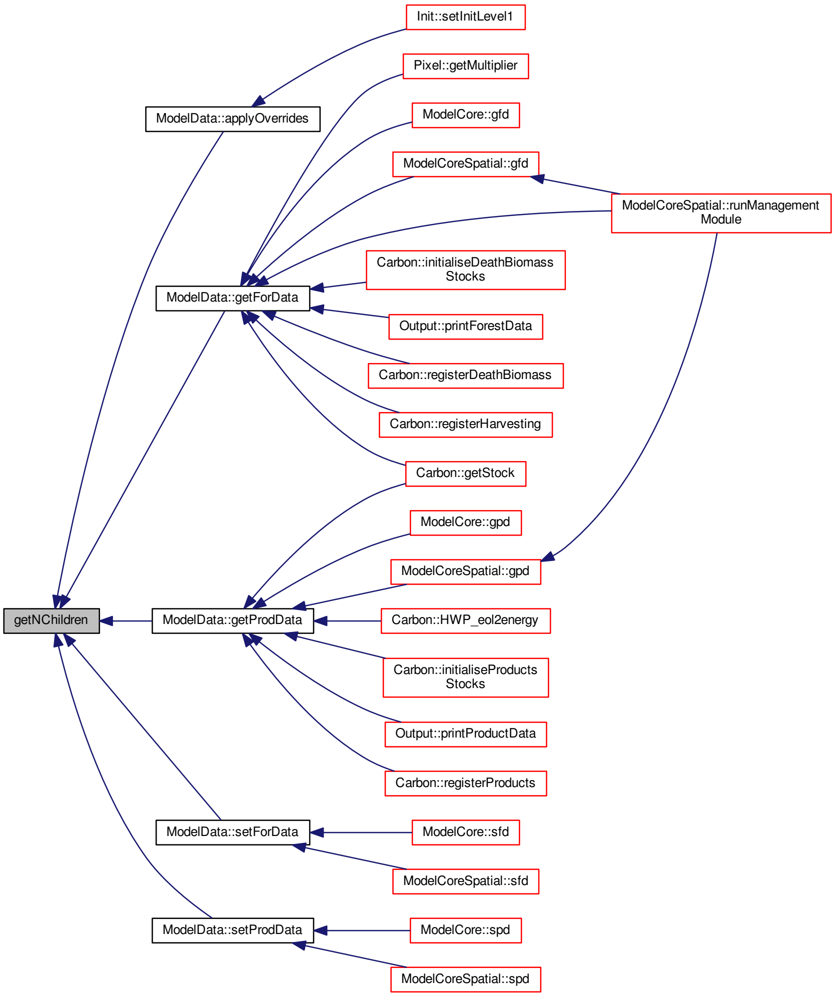
Referenced by ModelData::applyOverrides(), ModelData::getForData(), ModelData::getProdData(), ModelData::setForData(), and ModelData::setProdData().

```
00069                                           {
00070   if(excludeResidual){
00071     int toReturn;
00072     for(uint i=0;i<chRegions.size();i++){
00073       if(!chRegions[i]->getIsResidual()){
00074         toReturn++;
00075       }
00076     }
00077     return toReturn;
00078   }
00079   return chRegions.size();
00080 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



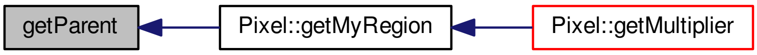
4.28.3.12 ModelRegion∗ getParent ( ) [inline]

Definition at line 72 of file ModelRegion.h.

Referenced by Pixel::getMyRegion().

```
00072 {return parRegion;}; ///< Returns a pointer to the parent regions
```

Here is the caller graph for this function:



4.28.3.13 int getParRegId ( ) const [inline]

Definition at line 70 of file ModelRegion.h.

```
00070 {return parRegId;};
```

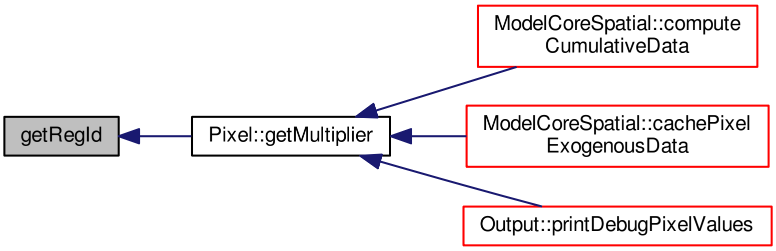
4.28.3.14 int getRegId ( ) const [inline]

Definition at line 66 of file ModelRegion.h.

Referenced by Pixel::getMultiplier().

```
00066 {return regId;};
```

Here is the caller graph for this function:


getRegId
Pixel::getMultiplier
ModelCoreSpatial::compute CumulativeData
ModelCoreSpatial::cachePixel ExogenousData
Output::printDebugPixelValues

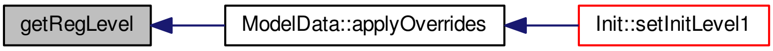
4.28.3.15 int getRegLevel ( ) const [inline]

Definition at line 69 of file ModelRegion.h.

Referenced by ModelData::applyOverrides().

```
00069 {return regLevel;};
```

Here is the caller graph for this function:



4.28.3.16 string getRegLName ( ) const [inline]

Definition at line 68 of file ModelRegion.h.

```
00068 {return regLName;};
```

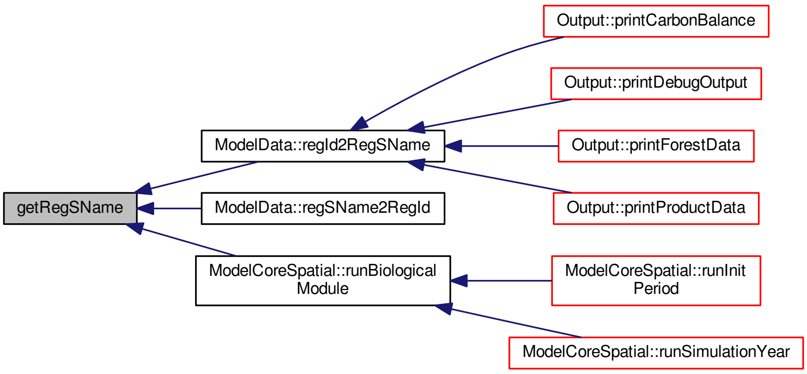
4.28.3.17 string getRegSName ( ) const [inline]

Definition at line 67 of file ModelRegion.h.

Referenced by ModelData::regId2RegSName(), ModelData::regSName2RegId(), and ModelCoreSpatial::run←BiologicalModule().

```
00067 {return regSName;};
```

Here is the caller graph for this function:


Output::printCarbonBalance
Output::printDebugOutput
ModelData::regId2RegSName
Output::printForestData
getRegSName
ModelData::regSName2RegId
Output::printProductData
ModelCoreSpatial::runBiological Module
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

4.28.3.18 double getValue (string *layerName,* int *op* = OP_SUM)

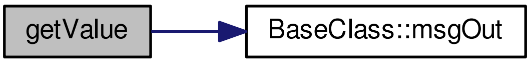
return the values of its own pixels for the specified layer. Possible operations: OP_SUM or OP_AVG

Definition at line 172 of file ModelRegion.cpp.

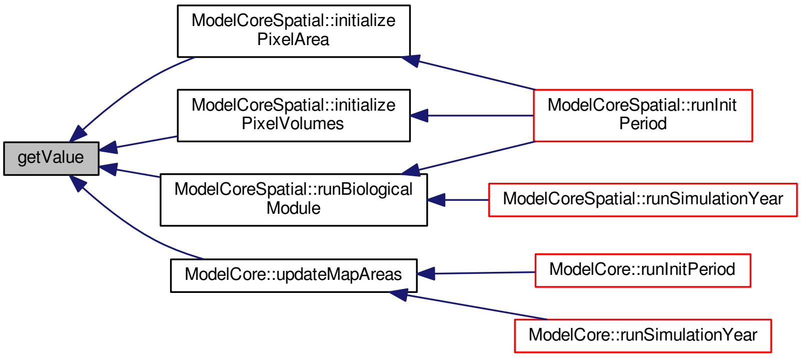
Referenced by ModelCoreSpatial::initializePixelArea(), ModelCoreSpatial::initializePixelVolumes(), ModelCoreSpatial::runBiologicalModule(), and ModelCore::updateMapAreas().

```
00172                                                {
00173   int nPx = myPixels.size();
00174   double sumvalue=0;
00175   for(uint i=0;i<nPx; i++){
00176     sumvalue += myPixels[i]->getDoubleValue(layerName,true);
00177   }
00178   if(op==OP_SUM){
00179     return sumvalue;
00180   } else if (op == OP_AVG) {
00181     return sumvalue/nPx;
00182   } else {
00183     string thisf = __PRETTY_FUNCTION__;
00184     msgOut(MSG_CRITICAL_ERROR, "in "+thisf+", operation not supported");
00185   }
00186   return 0.;
00187 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


ModelCoreSpatial::initialize PixelArea
ModelCoreSpatial::initialize PixelVolumes
getValue
ModelCoreSpatial::runBiological Module
ModelCore::updateMapAreas
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear
ModelCore::runInitPeriod
ModelCore::runSimulationYear

4.28.3.19 double getVolumes ( )

Todo Implement me (but really needed?)

Definition at line 85 of file ModelRegion.cpp.

```
00085                          {
00086   /// \todo Implement me (but really needed?)
00087   return 0;
00088 }
```

4.28.3.20 vector< double > getVolumes (int *fType_h*)

Todo Implement me (but really needed?)

Definition at line 91 of file ModelRegion.cpp.

```
00091                                     {
00092   /// \todo Implement me (but really needed?)
00093   vector<double> toReturn;
00094   return toReturn;
00095 }
```

4.28.3.21 vector< vector< double > > getVolumes (int *fType_h*, string *dClass_h*)

Todo Implement me (but really needed?)

Definition at line 98 of file ModelRegion.cpp.

```
00098                                                      {
00099   /// \todo Implement me (but really needed?)
00100   vector < vector <double> > toReturn;
00101   return toReturn;
00102 }
```

4.28.3.22 void setChildren (vector< **ModelRegion** ∗ > *children_h*) [inline]

Definition at line 59 of file ModelRegion.h.

```
00059 {chRegions = children_h;}; ///< Childrens are all the lvel-1 region that are parts of this region.
```

4.28.3.23 void setIsResidual (bool *isResidual_h*) [inline]

Definition at line 57 of file ModelRegion.h.

```
00057 {isResidual = isResidual_h;};
```

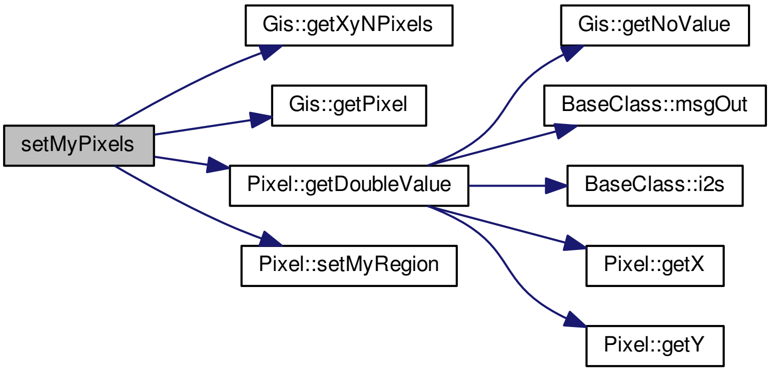
4.28.3.24 void setMyPixels ( )

It sets a double link pixels <–> region.

Definition at line 196 of file ModelRegion.cpp.

```
00196                        {
00197   int xyNPixels = MTHREAD->GIS->getXyNPixels();
00198   for(uint i=0;i<xyNPixels;i++){
00199     Pixel* px = MTHREAD->GIS->getPixel(i);
00200     if(px->getDoubleValue("regLev_1")==regId || px->
      getDoubleValue("regLev_2")==regId){
00201       myPixels.push_back(px);
00202       if(regLevel == 2){
00203         px->setMyRegion(this);
00204       }
00205     }
00206   }
00207 }
```

Here is the call graph for this function:



4.28.3.25 void setParent (**ModelRegion** ∗ *parRegion_h*) [inline]

Definition at line 58 of file ModelRegion.h.

```
00058 {parRegion = parRegion_h;};
```

4.28.3.26 void setParRegId (int *parRegId_h*) [inline]

Definition at line 56 of file ModelRegion.h.

```
00056 {parRegId = parRegId_h;};
```

4.28.3.27 void setRegId (int *regId_h*) [inline]

Definition at line 52 of file ModelRegion.h.

```
00052 {regId = regId_h;};
```

4.28.3.28 void setRegLevel (int *regLevel_h*) [inline]

Definition at line 55 of file ModelRegion.h.

```
00055 {regLevel = regLevel_h;};
```

4.28.3.29 void setRegLName (string *regLName_h*) [inline]

Definition at line 54 of file ModelRegion.h.

```
00054 {regLName = regLName_h;};
```

4.28.3.30 void setRegSName (string *regSName_h*) [inline]

Definition at line 53 of file ModelRegion.h.

```
00053 {regSName = regSName_h;};
```

4.28.3.31 void swap (const int & *swap_what*)

Definition at line 210 of file ModelRegion.cpp.

```
00210                                       {
00211
00212   for(uint i=0;i<myPixels.size();i++) {
00213     myPixels[i]->swap(swap_what);
00214   }
00215
00216 }
```

4.28.4 Member Data Documentation

4.28.4.1 vector<ModelRegion*> chRegions [private]

Vector of level-1 children regions.

Definition at line 98 of file ModelRegion.h.

Referenced by getChildren(), and getNChildren().

4.28.4.2 vector<forData*> forDataVector [private]

Vector of pointers of forestry data (owned by ModelData)

Definition at line 99 of file ModelRegion.h.

4.28.4.3 vector<double> inResByAnyCombination

Vector of inventory resource for each possible combination of primary products. This store both alive timber and death one.

Definition at line 85 of file ModelRegion.h.

Referenced by ModelCoreSpatial::computeInventary(), Opt::copyInventoryResourses(), ModelRegion(), and ModelCoreSpatial::runMarketModule().

4.28.4.4 vector<double> inResByAnyCombination_deathTimber

Vector of inventory resource for each possible combination of primary products. This store only death timber.

Definition at line 88 of file ModelRegion.h.

Referenced by ModelCoreSpatial::computeInventary(), and ModelRegion().

4.28.4.5 bool isResidual [private]

A flag if this region should be explicitelly modelled or it is just a residual.

Definition at line 96 of file ModelRegion.h.

Referenced by ModelRegion().

4.28.4.6 vector<Pixel*> myPixels [private]

Vector of pixels for this region.

Definition at line 101 of file ModelRegion.h.

Referenced by getArea(), getValue(), setMyPixels(), and swap().

4.28.4.7 int parRegId [private]

Id of the parent region;.

Definition at line 95 of file ModelRegion.h.

Referenced by ModelRegion().

4.28.4.8 ModelRegion* parRegion [private]

Pointer to the parent region.

Definition at line 97 of file ModelRegion.h.

4.28.4.9 vector<prodData*> prodDataVector [private]

Vector of pointers of product data (owned by ModelData)

Definition at line 100 of file ModelRegion.h.

4.28.4.10 int regId [private]

Regional unique ID.

Definition at line 91 of file ModelRegion.h.

Referenced by ModelRegion(), and setMyPixels().

4.28.4.11 int regLevel [private]

The level of the region. 1: country, 2: regions.

Definition at line 94 of file ModelRegion.h.

Referenced by ModelRegion(), and setMyPixels().

4.28.4.12 string regLName [private]

Region long name;.

Definition at line 93 of file ModelRegion.h.

Referenced by ModelRegion().

4.28.4.13 string regSName [private]

A short name of the region.

Definition at line 92 of file ModelRegion.h.

Referenced by ModelRegion().

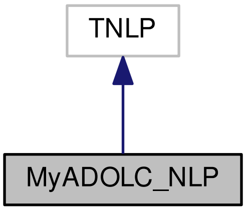
The documentation for this class was generated from the following files:

- /home/lobianco/git/ffsm_pp/src/ModelRegion.h
- /home/lobianco/git/ffsm_pp/src/ModelRegion.cpp

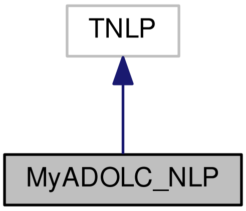
4.29 MyADOLC_NLP Class Reference

`#include <Adolc_debugtest.h>`

Inheritance diagram for MyADOLC_NLP:



Collaboration diagram for MyADOLC_NLP:



Public Member Functions

- MyADOLC_NLP ()
- virtual ∼MyADOLC_NLP ()
- virtual void generate_tapes (Index n, Index m)

Overloaded from TNLP

- virtual bool get_nlp_info (Index &n, Index &m, Index &nnz_jac_g, Index &nnz_h_lag, IndexStyleEnum &index_style)
- virtual bool get_bounds_info (Index n, Number ∗x_l, Number ∗x_u, Index m, Number ∗g_l, Number ∗g_u)
- virtual bool get_starting_point (Index n, bool init_x, Number ∗x, bool init_z, Number ∗z_L, Number ∗z_U, Index m, bool init_lambda, Number ∗lambda)
- template<class T >
  bool eval_obj (Index n, const T ∗x, T &obj_value)
- template<class T >
  bool eval_constraints (Index n, const T ∗x, Index m, T ∗g)

- virtual bool eval_f (Index n, const Number ∗x, bool new_x, Number &obj_value)
- virtual bool eval_grad_f (Index n, const Number ∗x, bool new_x, Number ∗grad_f)
- virtual bool eval_g (Index n, const Number ∗x, bool new_x, Index m, Number ∗g)
- virtual bool eval_jac_g (Index n, const Number ∗x, bool new_x, Index m, Index nele_jac, Index ∗iRow, Index ∗jCol, Number ∗values)
- virtual bool eval_h (Index n, const Number ∗x, bool new_x, Number obj_factor, Index m, const Number ∗lambda, bool new_lambda, Index nele_hess, Index ∗iRow, Index ∗jCol, Number ∗values)

Solution Methods

- virtual void finalize_solution (SolverReturn status, Index n, const Number ∗x, const Number ∗z_L, const Number ∗z_U, Index m, const Number ∗g, const Number ∗lambda, Number obj_value, const IpoptData ∗ip_data, IpoptCalculatedQuantities ∗ip_cq)

Methods to block default compiler methods.

- double ∗∗ Jac
- double ∗ x_lam
- double ∗∗ Hess
- MyADOLC_NLP (const MyADOLC_NLP &)
- MyADOLC_NLP & operator= (const MyADOLC_NLP &)

4.29.1 Detailed Description

Definition at line 37 of file Adolc_debugtest.h.

4.29.2 Constructor & Destructor Documentation

4.29.2.1 MyADOLC_NLP ( )

default constructor

Definition at line 34 of file Adolc_debugtest.cpp.

```
00035 {}
```

4.29.2.2 ∼MyADOLC_NLP ( ) `[virtual]`

default destructor

Definition at line 37 of file Adolc_debugtest.cpp.

```
00037 {}
```

4.29.2.3 MyADOLC_NLP (const MyADOLC_NLP &) [private]

4.29.3 Member Function Documentation

4.29.3.1 bool eval_constraints (Index *n*, const T ∗ *x*, Index *m*, T ∗ *g*)

Template to compute contraints

Definition at line 116 of file Adolc_debugtest.cpp.

```
{
  for (Index i=0; i<m; i++) {
    g[i] = 3.*pow(x[i+1],3.) + 2.*x[i+2] - 5.
           + sin(x[i+1]-x[i+2])*sin(x[i+1]+x[i+2]) + 4.*x[i+1]
           - x[i]*exp(x[i]-x[i+1]) - 3.;
  }

  return true;
}
```

4.29.3.2 bool eval_f (Index *n*, const Number ∗ *x*, bool *new_x*, Number & *obj_value*) [virtual]

Original method from Ipopt to return the objective value remains unchanged

Definition at line 136 of file Adolc_debugtest.cpp.

```
{
  eval_obj(n,x,obj_value);

  return true;
}
```

4.29.3.3 bool eval_g (Index *n*, const Number ∗ *x*, bool *new_x*, Index *m*, Number ∗ *g*) [virtual]

Original method from Ipopt to return the constraint residuals remains unchanged

Definition at line 151 of file Adolc_debugtest.cpp.

```
{

  eval_constraints(n,x,m,g);

  return true;
}
```

4.29.3.4 bool eval_grad_f (Index *n*, const Number ∗ *x*, bool *new_x*, Number ∗ *grad_f*) [virtual]

Original method from Ipopt to return the gradient of the objective remains unchanged

Definition at line 143 of file Adolc_debugtest.cpp.

```
{

  gradient(tag_f,n,x,grad_f);

  return true;
}
```

4.29.3.5 bool eval_h (Index *n*, const Number ∗ *x*, bool *new_x*, Number *obj_factor*, Index *m*, const Number ∗ *lambda*, bool *new_lambda*, Index *nele_hess*, Index ∗ *iRow*, Index ∗ *jCol*, Number ∗ *values*) [virtual]

Original method from Ipopt to return: 1) The structure of the hessian of the lagrangian (if "values" is NULL) 2) The values of the hessian of the lagrangian (if "values" is not NULL)remains unchanged

Definition at line 190 of file Adolc_debugtest.cpp.

```
{
  if (values == NULL) {
    // return the structure. This is a symmetric matrix, fill the lower left
    // triangle only.

    // the hessian for this problem is actually dense
    Index idx=0;
    for (Index row = 0; row < n; row++) {
      for (Index col = 0; col <= row; col++) {
        iRow[idx] = row;
        jCol[idx] = col;
        idx++;
      }
    }

    assert(idx == nele_hess);
  }
  else {
    // return the values. This is a symmetric matrix, fill the lower left
    // triangle only

    for(Index i = 0; i<n ; i++)
      x_lam[i] = x[i];
    for(Index i = 0; i<m ; i++)
      x_lam[n+i] = lambda[i];
    x_lam[n+m] = obj_factor;

    hessian(tag_L,n+m+1,x_lam,Hess);

    Index idx = 0;

    for(Index i = 0; i<n ; i++)
      {
    for(Index j = 0; j<=i ; j++)
      {
        values[idx++] = Hess[i][j];
      }
      }
  }

  return true;
}
```

4.29.3.6 bool eval_jac_g (Index *n*, const Number ∗ *x*, bool *new_x*, Index *m*, Index *nele_jac*, Index ∗ *iRow*, Index ∗ *jCol*, Number ∗ *values*) [virtual]

Original method from Ipopt to return: 1) The structure of the jacobian (if "values" is NULL) 2) The values of the jacobian (if "values" is not NULL)remains unchanged

Definition at line 159 of file Adolc_debugtest.cpp.

```
{
  if (values == NULL) {
    // return the structure of the jacobian,
    // assuming that the Jacobian is dense

    Index idx = 0;
    for(Index i=0; i<m; i++)
      for(Index j=0; j<n; j++)
    {
      iRow[idx] = i;
      jCol[idx++] = j;
    }
 }
  else {
    // return the values of the jacobian of the constraints
00177
00178     jacobian(tag_g,m,n,x,Jac);
00179
00180     Index idx = 0;
00181     for(Index i=0; i<m; i++)
00182       for(Index j=0; j<n; j++)
00183       values[idx++] = Jac[i][j];
00184
00185   }
00186
00187   return true;
00188 }
```

4.29.3.7 bool eval_obj (Index *n*, const T ∗ *x*, T & *obj_value*)

Template to return the objective value

Definition at line 103 of file Adolc_debugtest.cpp.

```
00104 {
00105   T a1, a2;
00106   obj_value = 0.;
00107   for (Index i=0; i<n-1; i++) {
00108     a1 = x[i]*x[i]-x[i+1];
00109     a2 = x[i] - 1.;
00110     obj_value += 100.*a1*a1 + a2*a2;
00111   }
00112
00113   return true;
00114 }
```

4.29.3.8 void finalize_solution (SolverReturn *status*, Index *n*, const Number ∗ *x*, const Number ∗ *z_L*, const Number ∗ *z_U*, Index *m*, const Number ∗ *g*, const Number ∗ *lambda*, Number *obj_value*, const IpoptData ∗ *ip_data*, IpoptCalculatedQuantities ∗ *ip_cq*) [virtual]

This method is called when the algorithm is complete so the TNLP can store/write the solution

Definition at line 237 of file Adolc_debugtest.cpp.

```
00243 {
00244
00245   printf("\n\nObjective value\n");
00246   printf("f(x*) = %e\n", obj_value);
00247
00248 // Memory deallocation for ADOL-C variables
00249
00250   delete[] x_lam;
00251
00252   for(Index i=0;i<m;i++)
00253     delete[] Jac[i];
00254   delete[] Jac;
00255
00256   for(Index i=0;i<n+m+1;i++)
00257     delete[] Hess[i];
00258   delete[] Hess;
00259 }
```

4.29.3.9 void generate_tapes (Index *n*, Index *m*) [virtual]

Method to generate the required tapes

Definition at line 264 of file Adolc_debugtest.cpp.

```
00265 {
00266   Number *xp    = new double[n];
00267   Number *lamp  = new double[m];
00268   Number *zl    = new double[m];
00269   Number *zu    = new double[m];
00270 
00271   adouble *xa   = new adouble[n];
00272   adouble *g    = new adouble[m];
00273   adouble *lam  = new adouble[m];
00274   adouble sig;
00275   adouble obj_value;
00276 
00277   double dummy;
00278 
00279   Jac = new double*[m];
00280   for(Index i=0;i<m;i++)
00281     Jac[i] = new double[n];
00282 
00283   x_lam   = new double[n+m+1];
00284 
00285   Hess = new double*[n+m+1];
00286   for(Index i=0;i<n+m+1;i++)
00287     Hess[i] = new double[i+1];
00288 
00289   get_starting_point(n, 1, xp, 0, zl, zu, m, 0, lamp);
00290 
00291   trace_on(tag_f);
00292 
00293     for(Index i=0;i<n;i++)
00294       xa[i] <<= xp[i];
00295 
00296     eval_obj(n,xa,obj_value);
00297 
00298     obj_value >>= dummy;
00299 
00300   trace_off();
00301 
00302   trace_on(tag_g);
00303 
00304     for(Index i=0;i<n;i++)
00305       xa[i] <<= xp[i];
00306 
00307     eval_constraints(n,xa,m,g);
00308 
00309 
00310     for(Index i=0;i<m;i++)
00311       g[i] >>= dummy;
00312 
00313   trace_off();
00314 
00315    trace_on(tag_L);
00316 
00317     for(Index i=0;i<n;i++)
00318       xa[i] <<= xp[i];
00319     for(Index i=0;i<m;i++)
00320       lam[i] <<= 1.0;
00321     sig <<= 1.0;
00322 
00323     eval_obj(n,xa,obj_value);
00324 
00325     obj_value *= sig;
00326     eval_constraints(n,xa,m,g);
00327 
00328     for(Index i=0;i<m;i++)
00329       obj_value += g[i]*lam[i];
00330 
00331     obj_value >>= dummy;
00332 
00333   trace_off();
00334 
00335   delete[] xa;
00336   delete[] xp;
00337   delete[] g;
00338   delete[] lam;
00339   delete[] lamp;
00340   delete[] zu;
00341   delete[] zl;
00342 
00343 }
```

4.29.3.10 bool get_bounds_info (Index *n*, Number ∗ *x_l*, Number ∗ *x_u*, Index *m*, Number ∗ *g_l*, Number ∗ *g_u*) `[virtual]`

Method to return the bounds for my problem

Definition at line 61 of file Adolc_debugtest.cpp.

```
{
  // none of the variables have bounds
  for (Index i=0; i<n; i++) {
    x_l[i] = -1e20;
    x_u[i] =  1e20;
  }

  // Set the bounds for the constraints
  for (Index i=0; i<m; i++) {
    g_l[i] = 0;
    g_u[i] = 0;
  }

  return true;
}
```

4.29.3.11 bool get_nlp_info (Index & *n*, Index & *m*, Index & *nnz_jac_g*, Index & *nnz_h_lag*, IndexStyleEnum & *index_style*) `[virtual]`

Method to return some info about the nlp

Definition at line 39 of file Adolc_debugtest.cpp.

```
{
  n = 20;

  m = n-2;

  // in this example the jacobian is dense. Hence, it contains n*m nonzeros
  nnz_jac_g = n*m;

  // the hessian is also dense and has n*n total nonzeros, but we
  // only need the lower left corner (since it is symmetric)
  nnz_h_lag = n*(n-1)/2+n;

  generate_tapes(n, m);

  // use the C style indexing (0-based)
  index_style = C_STYLE;

  return true;
}
```

4.29.3.12 bool get_starting_point (Index *n*, bool *init_x*, Number ∗ *x*, bool *init_z*, Number ∗ *z_L*, Number ∗ *z_U*, Index *m*, bool *init_lambda*, Number ∗ *lambda*) `[virtual]`

Method to return the starting point for the algorithm

Definition at line 79 of file Adolc_debugtest.cpp.

```
{
  // Here, we assume we only have starting values for x, if you code
  // your own NLP, you can provide starting values for the others if
  // you wish.
  assert(init_x == true);
  assert(init_z == false);
  assert(init_lambda == false);

  // set the starting point
  for (Index i=0; i<n/2; i++) {
    x[2*i] = -1.2;
    x[2*i+1] = 1.;
  }
  if (n != 2*(n/2)) {
    x[n-1] = -1.2;
  }

  return true;
}
```

4.29.3.13 MyADOLC_NLP& operator= (const MyADOLC_NLP &) `[private]`

4.29.4 Member Data Documentation

4.29.4.1 double** Hess `[private]`

Definition at line 141 of file Adolc_debugtest.h.

4.29.4.2 double** Jac `[private]`

Definition at line 138 of file Adolc_debugtest.h.

4.29.4.3 double* x_lam `[private]`

Definition at line 140 of file Adolc_debugtest.h.

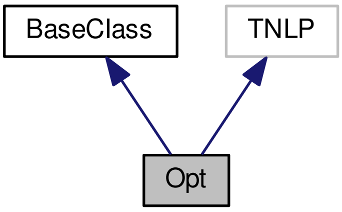
The documentation for this class was generated from the following files:

- /home/lobianco/git/ffsm_pp/src/Adolc_debugtest.h
- /home/lobianco/git/ffsm_pp/src/Adolc_debugtest.cpp

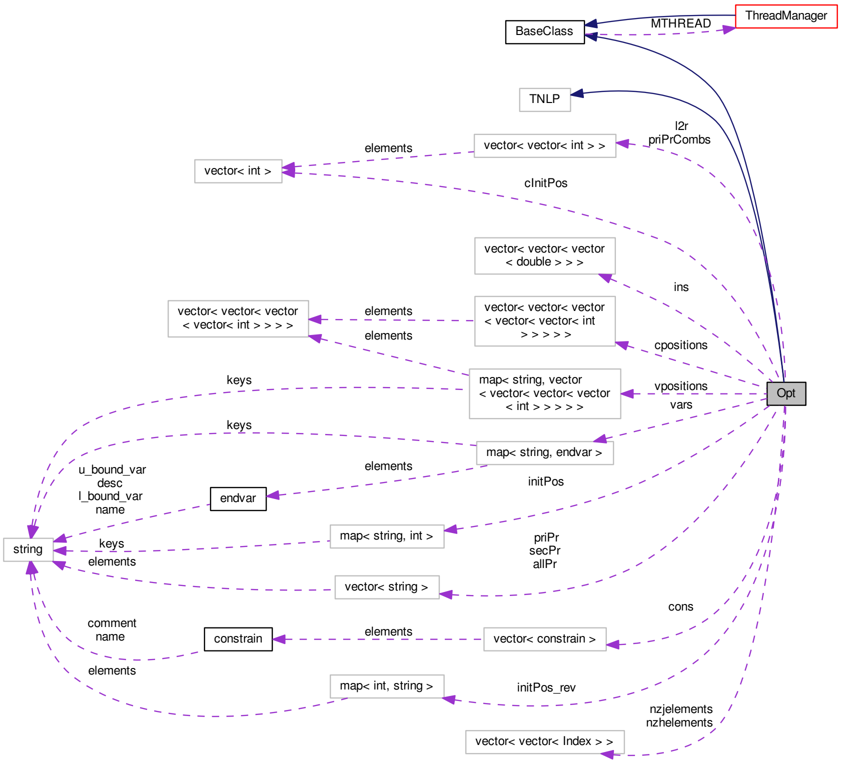
4.30 Opt Class Reference

`#include <Opt.h>`

Inheritance diagram for Opt:



Collaboration diagram for Opt:



Public Member Functions

- Opt (ThreadManager ∗MTHREAD_h)

  Constructor.
- ∼Opt ()
- virtual bool intermediate_callback (AlgorithmMode mode, Index iter, Number obj_value, Number inf_pr, Number inf_du, Number mu, Number d_norm, Number regularization_size, Number alpha_du, Number alpha_pr, Index ls_trials, const IpoptData ∗ip_data, IpoptCalculatedQuantities ∗ip_cq)
- virtual void generate_tapes (Index n, Index m, Index &nnz_jac_g, Index &nnz_h_lag)
- void declareVariables ()

  declare the variables, their domains and their bounds
- void declareVariable (const string &name, const int &domain, const string &desc="", const double &l_bound=0.0, const double &u_bound=UBOUND_MAX, const string &l_bound_var="", const string &u_bound_var="")

  Declare a single variable, its domain and its bounds.
- void declareConstrains ()

  declare the constrains, their domain, their direction and their associated evaluation function
- void cacheInitialPosition ()

  cache the initial positions of the variables and the constrains
- void calculateNumberVariablesConstrains ()

  calculate the number of variables and constrains
- void cachePositions ()

  cache the exact position index (initial+f(r1,r2,p,r2To) for each variable and constrain
- int getDomainElements (int domain)

return the number of elements of a domain

- template<class T >
  vector< vector< vector< vector< int > > > > buildPositionVector (const T &v_or_c, int dType)

  build the matrix of the positions for a given variable or contrain
- int getVarInstances (const string &varName)

  build the matrix of the positions for a given variable or contrain
- void calculateSparsityPatternJ ()
- void calculateSparsityPatternH ()
- const Number & mymax (const Number &a, const Number &b)
- const adouble & mymax (const adouble &a, const adouble &b)

Overloaded from TNLP

- virtual bool get_nlp_info (Index &n, Index &m, Index &nnz_jac_g, Index &nnz_h_lag, IndexStyleEnum &index_style)
- virtual bool get_bounds_info (Index n, Number ∗x_l, Number ∗x_u, Index m, Number ∗g_l, Number ∗g_u)
- virtual bool get_starting_point (Index n, bool init_x, Number ∗x, bool init_z, Number ∗z_L, Number ∗z_U, Index m, bool init_lambda, Number ∗lambda)
- template<class T >
  bool eval_obj (Index n, const T ∗x, T &obj_value)
- template<class T >
  bool eval_constraints (Index n, const T ∗x, Index m, T ∗g)
- virtual bool eval_f (Index n, const Number ∗x, bool new_x, Number &obj_value)
- virtual bool eval_grad_f (Index n, const Number ∗x, bool new_x, Number ∗grad_f)
- virtual bool eval_g (Index n, const Number ∗x, bool new_x, Index m, Number ∗g)
- virtual bool eval_jac_g (Index n, const Number ∗x, bool new_x, Index m, Index nele_jac, Index ∗iRow, Index ∗jCol, Number ∗values)
- virtual bool eval_h (Index n, const Number ∗x, bool new_x, Number obj_factor, Index m, const Number ∗lambda, bool new_lambda, Index nele_hess, Index ∗iRow, Index ∗jCol, Number ∗values)

Solution Methods

- virtual void finalize_solution (SolverReturn status, Index n, const Number ∗x, const Number ∗z_L, const Number ∗z_U, Index m, const Number ∗g, const Number ∗lambda, Number obj_value, const IpoptData ∗ip_data, IpoptCalculatedQuantities ∗ip_cq)

Protected Member Functions

- const double gpd (const string &type_h, const int &regId_h, const string &prodId_h, const int &year=DATA↩_NOW, const string &freeDim_h="") const
- const double gfd (const string &type_h, const int &regId_h, const string &forType_h, const string &diam↩Class_h, const int &year=DATA_NOW) const
- void spd (const double &value_h, const string &type_h, const int &regId_h, const string &prodId_h, const int &year=DATA_NOW, const bool &allowCreate=false, const string &freeDim_h="") const
- void sfd (const double &value_h, const string &type_h, const int &regId_h, const string &forType_h, const string &diamClass_h, const int &year=DATA_NOW, const bool &allowCreate=false) const
- bool app (const string &prod_h, const string &forType_h, const string &dClass_h) const
- const int gip (const string &varName) const

  Get the initial index position of a given variable in the concatenated array.
- const int gip (const int &cn) const

  Return the initial index position of a certain constrain.
- template<class T >
  const int gix_uncached (const T &v_or_c, int r1Ix, int r2Ix, int prIx, int r2IxTo=0)

  Get the index in the concatenated array gived a certain var name (string) or constrain index (int), the reg lev1 index, the reg lev2 index and the prod. index.

- const int gix (const string &varName, const int &r1Ix, const int &r2Ix, const int &prIx, const int &r2IxTo=0) const

  Get the index in the concatenated array gived a certain var name, the reg lev1 index, the reg lev2 index and the prod. index.
- const int gix (const int &cn, const int &r1Ix, const int &r2Ix, const int &prIx, const int &r2IxTo=0) const

  Get the index in the concatenated array gived a certain constrain, the reg lev1 index, the reg lev2 index and the prod. index.
- const int gdt (const string &varName)

  Get the domain type of a given variable.
- const int gdt (const int &cn)

  Get the domain type of a given constrain.
- int getConstrainDirectionByIndex (int idx)

  Return the direction of a given constrain.
- double getBoundByIndex (const int &bound_type, const int &idx)

  Return the bound of a given variable (by index)
- double getDetailedBoundByVarAndIndex (const endvar &var, const int &idx, const int &bType)

  Return the bound of a given variable given the variable and the required index. Called by getBoundByIndex().
- constrain ∗ getConstrainByIndex (int idx)
- void unpack (int ix_h, int domain, int initial, int &r1_h, int &r2_h, int &p_h, int &r2to_h, bool fullp=false)

  Return the dimensions given a certain index, domain type and initial position.
- int getConNumber (constrain ∗con)

  Return the position in the cons vector.
- void copyInventoryResourses ()

  Copy the inventoried resources in the in vector for better performances.
- void tempDebug ()
- void debugPrintParameters ()

Protected Attributes

- vector< string > priPr
- vector< string > secPr
- vector< string > allPr
- vector< vector< int > > l2r
- vector< vector< int > > priPrCombs

  A vector with all the possible combinations of primary products.
- vector< vector< vector< double > > > ins

  A copy of the inventoried resourses by region and primary product combination. It works also with dynamic loading of the region and the in, but it may be slower.
- map< string, int > initPos

  A map that returns the initial index position in the concatenated array for each variable.
- map< int, string > initPos_rev

  A map with the name of the variable keyed by its initial position in the index.
- vector< int > cInitPos

  A vector that returns the initial index position in the concatenated array for each constrain.
- map< string, endvar > vars

  List of variables in the model and their domain: pr product, sec prod, all products or all products over each subregion pair (exports)
- map< string, vector< vector< vector< vector< int > > > > > vpositions

  cached position in the concatenated vector for each variables. Dimensions are l1reg, l2reg, prod, (l2To region).
- vector< vector< vector< vector< vector< int > > > > > cpositions

  cached position in the concatenated vector for each variables. Dimensions are contrain number, l1reg, l2reg, prod, (l2To region).

- int nPriPr
- int nPriPrCombs
- int nSecPr
- int nAllPr
- int nL2r
- int nVar
- int nCons
- int nEqualityConstrains
- int nLowerEqualZeroConstrains
- int nGreaterEqualZeroConstrains
- int previousYear
- int firstYear
- int secondYear
- int worldCodeLev2
- bool debugRunOnce
- double overharvestingAllowance

  Allows to harvest more than the resources available. Useful when resources got completelly exausted and the model refuses to solve.
- bool initOpt
- vector< constrain > cons
- vector< vector< Index > > nzjelements

  nzero elements for the jacobian matrix. nzelements[i][0] -> row (constrain), nzelements[i][1] -> column (variable)
- vector< vector< Index > > nzhelements

  nzero elements for the hessian matrix

Methods to block default compiler methods.

- double ∗ x_lam
- unsigned int ∗∗ HP_t
- unsigned int ∗ rind_g
- unsigned int ∗ cind_g
- double ∗ jacval
- unsigned int ∗ rind_L
- unsigned int ∗ cind_L
- unsigned int ∗ rind_L_total
- unsigned int ∗ cind_L_total
- double ∗ hessval
- int nnz_jac
- int nnz_L
- int nnz_L_total
- int options_g [4]
- int options_L [4]
- Opt (const Opt &)
- Opt & operator= (const Opt &)

4.30.1 Detailed Description

Definition at line 52 of file Opt.h.

4.30.2 Constructor & Destructor Documentation

4.30.2.1 Opt (ThreadManager ∗ *MTHREAD_h*)

Constructor.

Definition at line 496 of file Opt.cpp.

```
00496                                  {
00497   MTHREAD = MTHREAD_h;
00498   nVar    = 0;
00499   nCons   = 0;
00500   debugRunOnce = false;
00501   initOpt = true;
00502 }
```

4.30.2.2 ∼Opt ( )

Definition at line 504 of file Opt.cpp.

```
00504           {
00505
00506 }
```

4.30.2.3 Opt (const Opt &) `[protected]`

4.30.3 Member Function Documentation

4.30.3.1 bool app (const string & *prod_h,* const string & *forType_h,* const string & *dClass_h*) const `[inline]`, `[protected]`

Definition at line 172 of file Opt.h.

```
00172 {return MTHREAD->MD->assessProdPossibility(prod_h, forType_h, dClass_h);};
```

4.30.3.2 vector< vector< vector< vector< int > > > > buildPositionVector (const T & *v_or_c,* int *dType*)

build the matrix of the positions for a given variable or contrain

Definition at line 1357 of file Opt.cpp.

```
01357                                                     {
01358   int pVectorSize;
01359
01360   switch (dType){
01361     case DOM_PRI_PR:
01362       pVectorSize= priPr.size();
01363       break;
01364     case DOM_SEC_PR:
01365       pVectorSize= secPr.size();
01366       break;
01367     case DOM_ALL_PR:
01368       pVectorSize= allPr.size();
01369       break;
01370     case DOM_R2_PRI_PR:
01371       pVectorSize= priPr.size();
01372       break;
01373     case DOM_R2_SEC_PR:
01374       pVectorSize= secPr.size();
01375       break;
01376     case DOM_R2_ALL_PR:
01377       pVectorSize= allPr.size();
01378       break;
01379     case DOM_SCALAR:
01380       pVectorSize= allPr.size(); // it will simply fill the matrix all with the same value (the ip)
01381       break;
01382     case DOM_PRI_PR_ALLCOMBS:
01383       pVectorSize= priPrCombs.size();
01384       break;
01385     default:
01386       msgOut(MSG_CRITICAL_ERROR,"Try to build the position of a variable (or
       contrain) of unknow type.");
01387   }
01388 
01389 
01390   vector < vector < vector < vector <int> > > > positionsToAdd;
01391   for(uint r1=0;r1<l2r.size();r1++){
01392     vector < vector < vector <int> > > dim1;
01393     for(uint r2=0;r2<l2r[r1].size();r2++){
01394       vector < vector <int> > dim2;
01395       for(uint p=0;p<pVectorSize;p++){
01396         vector <int> dim3;
01397           for(uint r2To=0;r2To<l2r[r1].size();r2To++){
01398             dim3.push_back(gix_uncached(v_or_c,r1,r2,p,r2To));
01399           }
01400         dim2.push_back(dim3);
01401       }
01402       dim1.push_back(dim2);
01403     }
01404     positionsToAdd.push_back(dim1);
01405   }
01406   return positionsToAdd;
01407 }
```

4.30.3.3 void cacheInitialPosition ( )

cache the initial positions of the variables and the constrains

Definition at line 1326 of file Opt.cpp.

```
01326                           {
01327   int vInitialPosition = 0;
01328   int cInitialPosition = 0;
01329   VarMap::iterator viter;
01330   for (viter = vars.begin(); viter != vars.end(); ++viter) {
01331     initPos.insert(pair<string, int>(viter->first, vInitialPosition));
01332     initPos_rev.insert(pair<int, string>(vInitialPosition,viter->first));
01333     vInitialPosition += getDomainElements(viter->second.domain);
01334   }
01335   for (uint i=0;i<cons.size();i++){
01336     cInitPos.push_back(cInitialPosition);
01337     cInitialPosition += getDomainElements(cons[i].domain);
01338   }
01339 }
```

4.30.3.4 void cachePositions ( )

cache the exact position index (initial+f(r1,r2,p,r2To) for each variable and constrain

Definition at line 1342 of file Opt.cpp.

```
01342                     {
01343 
01344   // variables..
01345   VarMap::iterator viter;
01346   for (viter = vars.begin(); viter != vars.end(); ++viter) {
01347     vpositions.insert(pair<string, vector < vector < vector < vector <int> > > > >(viter->first,
       buildPositionVector(viter->first, viter->second.domain)));
01348   }
01349   // constrains..
01350   for (uint i=0; i<cons.size();i++){
01351     cpositions.push_back(buildPositionVector(i,
      cons[i].domain));
01352   }
01353 
01354 }
```

4.30.3.5 void calculateNumberVariablesConstrains ( )

calculate the number of variables and constrains

Definition at line 1411 of file Opt.cpp.

```
01411                                        {
01412     // calculating the number of variables and the initial positions in the concatenated array..
01413     nVar = 0;
01414     VarMap::iterator viter;
01415     for (viter = vars.begin(); viter != vars.end(); ++viter) {
01416         nVar += getDomainElements(viter->second.domain);
01417     }
01418 
01419     // calculating the number of constrains..
01420     nCons = 0;
01421     nEqualityConstrains = 0;
01422     nLowerEqualZeroConstrains = 0;
01423     nGreaterEqualZeroConstrains = 0;
01424     for(uint i=0;i<cons.size();i++){
01425       nCons += getDomainElements(cons[i].domain);
01426       if(cons[i].direction == CONSTR_EQ){
01427         nEqualityConstrains += getDomainElements(
      cons[i].domain);
01428         continue;
01429       } else if (cons[i].direction == CONSTR_LE0) {
01430         nLowerEqualZeroConstrains += getDomainElements(
      cons[i].domain);
01431         continue;
01432       } else if (cons[i].direction == CONSTR_GE0) {
01433         nGreaterEqualZeroConstrains +=
      getDomainElements(cons[i].domain);
01434         continue;
01435       } else {
01436         msgOut(MSG_CRITICAL_ERROR, "Asking for a constrain with unknown direction (
      "+i2s(cons[i].direction)+")");
01437       }
01438     }
01439 
01440     msgOut(MSG_INFO,"The model will work with "+i2s(nVar)+" variables and "+
      i2s(nCons)+" constrains ("+i2s(nEqualityConstrains)+" equalities, "+
      i2s(nLowerEqualZeroConstrains)+" lower than 0 and "+
      i2s(nGreaterEqualZeroConstrains)+" greater than 0)");
01441 }
```

4.30.3.6 void calculateSparsityPatternH ( )

Definition at line 1664 of file Opt.cpp.

```
01664                               {
01665 
01666   unsigned int  **hesspat=NULL; // compressed row storage
01667   int           options_h=0; // options for the hessian patterns
01668   double        *x;
01669   int retv_h = -1; // return value
01670 
01671   hesspat = new unsigned int* [(nVar+nCons+1)];
01672   x = new double[(nVar+nCons+1)];
01673 
01674   retv_h = hess_pat(tag_L,nVar+nCons+1,  x, hesspat, options_h);
01675 
01676   for (int i=0;i<(nVar);i++) {
01677     for (int j=1;j<=hesspat[i][0];j++){
01678       if(hesspat[i][j]<=i){
01679         vector <int> nzhelement;
01680         nzhelement.push_back(i);
01681         nzhelement.push_back(hesspat[i][j]);
01682         nzhelements.push_back(nzhelement);
01683       }
01684     }
01685   }
01686 }
```

4.30.3.7 void calculateSparsityPatternJ ( )

Definition at line 1636 of file Opt.cpp.

```
01636                               {
01637
01638   unsigned int  **jacpat=NULL; // compressed row storage
01639   int           options_j[3]; // options for the jacobian patterns
01640   double        *x;
01641   int retv_j = -1; // return value
01642
01643   options_j[0] = 0; // index domain propagation
01644   options_j[1] = 0; // automatic mode choice (ignored here)
01645   options_j[2] = 0; // safe
01646   jacpat = new unsigned int* [nCons];
01647   x = new double[nVar];
01648
01649   nzjelements.clear();
01650
01651   retv_j = jac_pat(tag_g, nCons, nVar,  x, jacpat, options_j);
01652
01653   for (int i=0;i<nCons;i++) {
01654     for (int j=1;j<=jacpat[i][0];j++){
01655       vector <int> nzjelement;
01656       nzjelement.push_back(i);
01657       nzjelement.push_back(jacpat[i][j]);
01658       nzjelements.push_back(nzjelement);
01659     }
01660   }
01661 }
```

4.30.3.8 void copyInventoryResourses ( ) [protected]

Copy the inventoried resources in the in vector for better performances.

Opt::createCombinationsVector Return a vector containing any possible combination of nItems items (including all subsets).

For example with nItems = 3: 0: []; 1: [0]; 2: [1]; 3: [0,1]; 4: [2]; 5: [0,2]; 6: [1,2]; 7: [0,1,2]

Parameters

nItems	number of items to create p

Returns

A vector with in each slot the items present in that specific combination subset.

Definition at line 1801 of file Opt.cpp.

```
01801                            {
01802   // This function is not really needed, as actually the solver works also picking the region and the in
       dynamically
01803   // Caching the inventories in a vector should however be faster.
01804   // We now need it, as the vector inResByAnyCombination() account for the union between the inv set of the
       various pp. Also it now include the total mortality (alive plus death, if modelled)
01805   vector < vector < vector <double> > >  in_temp;
01806   for (uint r1=0;r1<l2r.size();r1++){
01807      vector < vector <double> > dim1;
01808     for (uint r2=0;r2<l2r[r1].size();r2++){
01809       vector <double> dim2;
01810       ModelRegion* REG = MTHREAD->MD->getRegion(l2r[r1][r2]);
01811       for (uint p=0;p<priPrCombs.size();p++){
01812         double this_in = REG->inResByAnyCombination[p];
01813         dim2.push_back(this_in);
01814       }
01815       dim1.push_back(dim2);
01816     }
01817     in_temp.push_back(dim1);
01818   }
01819   ins = in_temp;
01820 }
```

4.30.3.9 void debugPrintParameters ( ) [protected]

4.30.3.10 void declareConstrains ( )

declare the constrains, their domain, their direction and their associated evaluation function

Declare the constrains and their properties. For the domain type

See also

BaseClass

Definition at line 84 of file Opt.cpp.

```
00084                            {
00085   // domain of constrains variables
00086   // for domain
00087   constrain mkeq2;
00088   mkeq2.name="mkeq2";
00089   mkeq2.comment="[h1] Conservation of matters of transformed products";
00090   mkeq2.domain=DOM_SEC_PR;
00091   mkeq2.direction = CONSTR_EQ;
00092   //mkeq2.evaluate = Opt::mkteq2f;
00093
00094   constrain mkeq3;
00095   mkeq3.name="mkeq3";
00096   mkeq3.comment="[h2] Conservation of matters of raw products";
00097   mkeq3.domain=DOM_PRI_PR;
00098   mkeq3.direction = CONSTR_EQ;
00099   //mkeq3.evaluate = Opt::mkteq3f;
00100
00101   constrain mkeq4;
00102   mkeq4.name="mkeq4";
00103   mkeq4.comment="[eq 13] Leontief transformation function";
00104   mkeq4.domain=DOM_PRI_PR;
00105   mkeq4.direction = CONSTR_EQ;
00106
00107   constrain mkeq5;
00108   mkeq5.name="mkeq5";
00109   mkeq5.comment="[eq 21] Raw product supply function";
00110   mkeq5.domain=DOM_PRI_PR;
00111   mkeq5.direction = CONSTR_EQ;
00112
00113   constrain mkeq6;
00114   mkeq6.name="mkeq6";
00115   mkeq6.comment="[eq 20] Trasformed products demand function";
00116   mkeq6.domain=DOM_SEC_PR;
00117   mkeq6.direction = CONSTR_EQ;
00118
00119   constrain mkeq7;
00120   mkeq7.name="mkeq7";
00121   mkeq7.comment="[h7 and h3] Transformed products import function";
00122   mkeq7.domain=DOM_SEC_PR;
00123   mkeq7.direction = CONSTR_EQ;
00124
00125   constrain mkeq8;
00126   mkeq8.name="mkeq8";
00127   mkeq8.comment="[h8 and h4] Raw products export function";
00128   mkeq8.domain=DOM_PRI_PR;
00129   mkeq8.direction = CONSTR_EQ;
00130
00131   constrain mkeq13;
00132   mkeq13.name="mkeq13";
00133   mkeq13.comment="[h9] Calculation of the composite price of transformed products (PPC_Dp)";
00134   mkeq13.domain=DOM_SEC_PR;
00135   mkeq13.direction = CONSTR_EQ;
00136
00137   constrain mkeq14;
00138   mkeq14.name="mkeq14";
```

```
 mkeq14.comment="[h10] Calculation of the composite price of raw products (PPC_Sw)";
 mkeq14.domain=DOM_PRI_PR;
 mkeq14.direction = CONSTR_EQ;

 constrain mkeq17;
 mkeq17.name="mkeq17";
 mkeq17.comment="[h16] Constrain of the transformaton supply (lower than the regional maximal production capacity)";
 mkeq17.domain=DOM_SEC_PR;
 mkeq17.direction = CONSTR_LE0;


 constrain mkeq23;
 mkeq23.name="mkeq23";
 mkeq23.comment="[h3] Composit demand eq. (Dp)";
 mkeq23.domain=DOM_SEC_PR;
 mkeq23.direction = CONSTR_EQ;

 constrain mkeq24;
 mkeq24.name="mkeq24";
 mkeq24.comment="[h4] Composite supply eq. (Sw)";
 mkeq24.domain=DOM_PRI_PR;
 mkeq24.direction = CONSTR_EQ;

 constrain mkeq26;
 mkeq26.name="mkeq26";
 mkeq26.comment="[eq ] Verification of the null transport agents supply";
 mkeq26.domain=DOM_R2_ALL_PR;
 mkeq26.direction = CONSTR_LE0;

 constrain mkeq25;
 mkeq25.name="mkeq25";
 mkeq25.comment="Verification of the null trasformers supply (price of raw product + trasf product > trasf product)";
 mkeq25.domain=DOM_SEC_PR;
 mkeq25.direction = CONSTR_GE0;

 constrain mkeq18;
 mkeq18.name="mkeq18";
 mkeq18.comment="Constrain on raw material supply (lower than inventory)";
 mkeq18.domain=DOM_PRI_PR;
 mkeq18.direction = CONSTR_LE0;

 constrain resbounds;
 resbounds.name="resbounds";
 resbounds.comment="Constrain on raw material supply (lower than inventory, for each possible combination of primary products)";
 resbounds.domain=DOM_PRI_PR_ALLCOMBS;
 resbounds.direction = CONSTR_LE0;



 //constrain steq;
 //steq.name="steq";
 //steq.comment="computation of total supply";
 //steq.domain=DOM_PRI_PR;
 //steq.direction = CONSTR_EQ;

 cons.push_back(mkeq2);
 cons.push_back(mkeq6);
 cons.push_back(mkeq7);
 cons.push_back(mkeq13);
 cons.push_back(mkeq23);
 cons.push_back(mkeq3);
 cons.push_back(mkeq4);
 cons.push_back(mkeq5);
 cons.push_back(mkeq8);
 cons.push_back(mkeq14);
 cons.push_back(mkeq24);
 cons.push_back(mkeq17);
 cons.push_back(mkeq26);
 cons.push_back(mkeq25);
 //cons.push_back(mkeq18);
 cons.push_back(resbounds);
 //cons.push_back(steq);
;



}
```

4.30.3.11 void declareVariable (const string & *name,* const int & *domain,* const string & *desc* = "", const double & *l_bound* = 0.0, const double & *u_bound* = UBOUND_MAX, const string & *l_bound_var* = "", const string & *u_bound_var* = "")

Declare a single variable, its domain and its bounds.

Opt::declareVariable Define a single variable together with its domain and optionally its lower and upper bound (default 0.0, +inf)

Parameters

name	var name
domain	domain of the variable
l_bound	lower bound (fixed)
u_bound	upper bound (fixed)
l_bound_var	variable name defining lower bound
u_bound_var	variable name defining upper bound

Definition at line 1747 of file Opt.cpp.

```
01747
                                                                                  {
01748    endvar end_var;
01749    end_var.name = name;
01750    end_var.domain = domain;
01751    end_var.l_bound = l_bound;
01752    end_var.u_bound = u_bound;
01753    end_var.l_bound_var = l_bound_var;
01754    end_var.u_bound_var = u_bound_var;
01755    end_var.desc= desc;
01756    vars.insert(std::pair<std::string, endvar >(name, end_var));
01757 }
```

4.30.3.12 void declareVariables ( )

declare the variables, their domains and their bounds

Definition at line 59 of file Opt.cpp.

```
00059                     {
00060     // filling the list of variables and their domain and optionally their bonds
00061     // if you add variables in the model that enter optimisation you'll have to add them here
00062     // the underlying map goes automatically in alphabetical order
00063     // original order: pc,pl,dc,dl,da,sc,sl,sa,exp
00064     // 20140328: if these vars have a lower bound > 0 the model doesn't solve when volumes in a region go
      to zero !!!
00065
00066     // syntax: declareVariable("name", domainType, lbound[default=0], ubound[default= +inf], variable
      defining lower bounds[default=""], variable defining upper bound[default=""])
00067
00068     // all variables have upper or equal than zero bound:
00069     declareVariable("da", DOM_SEC_PR,    "Demand from abroad (imports)");
00070     declareVariable("dc", DOM_SEC_PR,    "Demand, composite");
00071     declareVariable("dl", DOM_ALL_PR,    "Demand from local");
00072     declareVariable("pc", DOM_ALL_PR,    "Price, composite");
00073     declareVariable("pl", DOM_ALL_PR,    "Price, local") ;
00074     declareVariable("rt", DOM_R2_ALL_PR, "Regional trade"); //it was exp in
      gams
00075     declareVariable("sa", DOM_PRI_PR,    "Supply to abroad (exports)");
00076     declareVariable("sc", DOM_PRI_PR,    "Supply, composite");
00077     declareVariable("sl", DOM_ALL_PR,    "Supply to locals");
00078     //declareVariable("st", DOM_PRI_PR,    "Supply, total", 0.0,UBOUND_MAX,"","in");
00079 }
```

4.30.3.13 bool eval_constraints (Index *n*, const T ∗ *x*, Index *m*, T ∗ *g*)

Template to compute contraints

Template function to implement (define) the previously declared constains. To the initial macro loop it must be passed the product vector over where to loop (priPr, secPr or allPr) and the order of the constrain has it has been added to the const vector. It could be possible to change this in a map and uses name, but then we would loose control on the constrains order, and we saw that it matters for finding the equilibrium.

Definition at line 305 of file Opt.cpp.

```
00305                                                          {
00306
00307   double a_pr, a, sigma, ff, sub_s, sub_d, sub_d_pSubstituted, sub_d_1, sub_d_1_pSubstituted, gg, q1, p1v,
      t1, r1v, psi, eta, pworld, ct, k, dispor, mv, in, in_1, supCorr, es_d, pc_1, pc_1_pSubstituted;
00308   Index cix = 0;
00309   Index debug = 0;
00310
00311   // mkteq2(i,p_tr)..  RVAR('dl',i,p_tr)+sum(j,EXP(i,j,p_tr))  =e=  RVAR('sl',i,p_tr)+
       sum(b,EXP(b,i,p_tr)); // h1
00312   CONSTRAIN_START_LOOP(secPr, 0) // attenction! you have to give the same order
       number as you inserted in the cons vector
00313     //g[cix] = x[gix("dl",r1,r2,psec)]-x[gix("sl",r1,r2,psec)]+x[gix("da",r1,r2,p)];
00314     g[cix] = x[gix("dl",r1,r2,psec)]-x[gix("sl",r1,r2,psec)];
00315     for (uint r2To=0;r2To<l2r[r1].size();r2To++){
00316       g[cix] += x[gix("rt",r1,r2,psec,r2To)]-x[gix("rt",r1,r2To,psec,r2)];
00317     }
00318   CONSTRAIN_END_LOOP
00319
00320   // mkteq6(i,p_tr)..  RVAR('dc',i,p_tr)  =e=  GG(i,p_tr)*(RVAR('pc',i,p_tr)**sigma(p_tr)); // eq. 20
       20160216: added sustitution elasticity in the demand
00321   // DEMAND EQUATION of transformed products
00322   CONSTRAIN_START_LOOP(secPr,1)
00323     gg      = gpd("gg",l2r[r1][r2],secPr[p]);
00324     sigma   = gpd("sigma",l2r[r1][r2],secPr[p]);
00325     pc_1    = gpd("pc",l2r[r1][r2],secPr[p],previousYear);
00326     sub_d   = gpd("sub_d",l2r[r1][r2],secPr[p]);              // subside this year
00327     sub_d_1 = gpd("sub_d",l2r[r1][r2],secPr[p],previousYear); // subside previous year
00328     g[cix]  = - gg*pow(x[gix("pc",r1,r2,psec)],sigma);
00329     for (uint p2=0;p2<secPr.size();p2++){
00330       es_d     = gpd("es_d",l2r[r1][r2],secPr[p],DATA_NOW,
      secPr[p2]);
00331       pc_1_pSubstituted    = gpd("pc",l2r[r1][r2],secPr[p2],previousYear);
00332       sub_d_pSubstituted   = gpd("pc",l2r[r1][r2],secPr[p2]);               // subside this year
       for the substitute product
00333       sub_d_1_pSubstituted = gpd("pc",l2r[r1][r2],secPr[p2],previousYear);  // subside last year
       for the substitute product
00334
00335       g[cix] *= pow(
00336                 (
00337                   ((x[gix("pc",r1,r2,psec)]+sub_d) / (x[gix("pc",r1,r2,
      priPr.size()+p2)]+sub_d_pSubstituted))
00338                        /
00339                   ((pc_1+sub_d_1) / (pc_1_pSubstituted+sub_d_1_pSubstituted))
00340                 ), es_d
00341               );
00342     }
00343     //g[cix] = x[gix("dc",r1,r2,p)]-gg*pow(x[gix("pc",r1,r2,psec)],sigma); // original without substitution
       elasticity
00344     g[cix] += x[gix("dc",r1,r2,p)];
00345   CONSTRAIN_END_LOOP
00346
00347   // mkteq7(i,p_tr)..  RVAR('da',i,p_tr)/RVAR('dl',i,p_tr)  =e=
       ((q1(i,p_tr)*RVAR('pl',i,p_tr))/(p1(i,p_tr)*PT_t(p_tr)))**psi(i,p_tr); // h7 and h3 ?
00348   CONSTRAIN_START_LOOP(secPr,2)
00349     q1 = gpd("q1",l2r[r1][r2],secPr[p]);
00350     p1v = 1-q1;
00351     psi = gpd("psi",l2r[r1][r2],secPr[p]);
00352     pworld = gpd("pl", worldCodeLev2,secPr[p]);
00353     g[cix] = x[gix("da",r1,r2,p)]/x[gix("dl",r1,r2,psec)] - pow((q1*x[gix("pl",r1,r2,psec)])/(p1v*
      pworld),psi);
00354   CONSTRAIN_END_LOOP
00355
00356   // mkteq13(i,p_tr).. RVAR('pc',i,p_tr)*RVAR('dc',i,p_tr)  =e=
       RVAR('dl',i,p_tr)*RVAR('pl',i,p_tr)+RVAR('da',i,p_tr)*PT_t(p_tr);  // h9
00357   CONSTRAIN_START_LOOP(secPr,3)
00358     pworld = gpd("pl", worldCodeLev2,secPr[p]);
00359     g[cix] = x[gix("pc",r1,r2,psec)]*x[gix("dc",r1,r2,p)]-x[gix("dl",r1,r2,psec)]*x[
      gix("pl",r1,r2,psec)]-x[gix("da",r1,r2,p)]*pworld;
00360   CONSTRAIN_END_LOOP
00361
```

```
 // mkteq23(i,p_tr).. RVAR('dc',i,p_tr)  =e=
(q1(i,p_tr)*(RVAR('da',i,p_tr)**((psi(i,p_tr)-1)/psi(i,p_tr)))+ p1(i,p_tr)*(RVAR('dl',i,p_tr)**((psi(i,p_tr)-1)/psi(i,p
 CONSTRAIN_START_LOOP(secPr,4)
   q1 = gpd("q1",l2r[r1][r2],secPr[p]);
   psi = gpd("psi",l2r[r1][r2],secPr[p]);
   p1v = 1-q1;
   g[cix] = x[gix("dc",r1,r2,p)] -
     pow(
           q1 * pow(x[gix("da",r1,r2,p)],(psi-1)/psi)
         + p1v * pow(x[gix("dl",r1,r2,psec)],(psi-1)/psi),
         psi/(psi-1)
     );
 CONSTRAIN_END_LOOP

 // mkteq3(i,p_pr)..  RVAR('dl',i,p_pr)+sum(j,EXP(i,j,p_pr))   =e=  RVAR('sl',i,p_pr)+
sum(b,EXP(b,i,p_pr))+sum(p_pr2, pres(p_pr2,p_pr)* RVAR('sl',i,p_pr2)); // h2
 CONSTRAIN_START_LOOP(priPr,5)
   //g[cix] = x[gix("dl",r1,r2,p)]-x[gix("sl",r1,r2,p)]-x[gix("sa",r1,r2,p)];
   g[cix] = x[gix("dl",r1,r2,p)]-x[gix("sl",r1,r2,p)];
   for (uint r2To=0;r2To<l2r[r1].size();r2To++){
     g[cix] += x[gix("rt",r1,r2,p,r2To)]-x[gix("rt",r1,r2To,p,r2)];
   }
   for (uint p2=0;p2<priPr.size();p2++){
     a_pr = gpd("a_pr",l2r[r1][r2],priPr[p2],DATA_NOW,priPr[p]);
     g[cix] -= a_pr*x[gix("sl",r1,r2,p2)];
   }
 CONSTRAIN_END_LOOP

 //mkteq4(i,p_pr)..  RVAR('dl',i,p_pr)  =e=  sum(p_tr, a(p_pr,p_tr)*(RVAR('sl',i,p_tr))); // eq. 13
 CONSTRAIN_START_LOOP(priPr,6)
   g[cix] = x[gix("dl",r1,r2,p)];
   for (uint p2=0;p2<secPr.size();p2++){
     a = gpd("a",l2r[r1][r2],priPr[p],DATA_NOW,secPr[p2]);
     g[cix] -= a*x[gix("sl",r1,r2,p2+nPriPr)];
   }
 CONSTRAIN_END_LOOP

 // mkteq5(i,p_pr)..  RVAR('sc',i,p_pr)  =e=  FF(i,p_pr)*(RVAR('pc',i,p_pr)**sigma(p_pr)); // eq. 21
 // SUPPLY EQUATION OF PRIMARY PRODUCTS
 CONSTRAIN_START_LOOP(priPr,7)
   ff = gpd("ff",l2r[r1][r2],priPr[p]);
   sub_s = gpd("sub_s",l2r[r1][r2],priPr[p]);
   sigma = gpd("sigmaCorr",l2r[r1][r2],priPr[p]);
   //g[cix] = x[gix("sc",r1,r2,p)]-mymax(ff*pow(x[gix("pc",r1,r2,p)],sigma),0.001);
   g[cix] = x[gix("sc",r1,r2,p)]-ff*pow(x[gix("pc",r1,r2,p)]+sub_s,sigma);
   //g[cix] = x[gix("sc",r1,r2,p)]-ff*pow(x[gix("pc",r1,r2,p)],sigma-0.0001);
 CONSTRAIN_END_LOOP


 // mkteq8(i,p_pr)..  RVAR('sa',i,p_pr)/RVAR('sl',i,p_pr)  =e=
((t1(i,p_pr)*RVAR('pl',i,p_pr))/(r1(i,p_pr)*PT_t(p_pr)))**eta(i,p_pr); // h8 and h4 ?
 CONSTRAIN_START_LOOP(priPr,8)
   t1 = gpd("t1",l2r[r1][r2],priPr[p]);
   r1v = 1-t1;
   eta = gpd("eta",l2r[r1][r2],priPr[p]);
   pworld = gpd("pl", worldCodeLev2,priPr[p]);
   g[cix] = x[gix("sa",r1,r2,p)]/x[gix("sl",r1,r2,p)] - pow((t1*x[gix("pl",r1,r2,p)])/(r1v*pworld
),eta);
 CONSTRAIN_END_LOOP

 // mkteq14(i,p_pr).. RVAR('pc',i,p_pr)*RVAR('sc',i,p_pr)  =e=
RVAR('sl',i,p_pr)*RVAR('pl',i,p_pr)+RVAR('sa',i,p_pr)*PT_t(p_pr);  // h10
 CONSTRAIN_START_LOOP(priPr,9)
   pworld = gpd("pl", worldCodeLev2,priPr[p]);
   g[cix] = x[gix("pc",r1,r2,p)]*x[gix("sc",r1,r2,p)]-x[gix("sl",r1,r2,p)]*x[
gix("pl",r1,r2,p)]-x[gix("sa",r1,r2,p)]*pworld;
 CONSTRAIN_END_LOOP

 //mkteq24(i,p_pr).. RVAR('sc',i,p_pr)  =e=
(t1(i,p_pr)*(RVAR('sa',i,p_pr)**((eta(i,p_pr)-1)/eta(i,p_pr)))+ r1(i,p_pr)*(RVAR('sl',i,p_pr)**((eta(i,p_pr)-1)/eta(i,p
 CONSTRAIN_START_LOOP(priPr,10)
   t1 = gpd("t1",l2r[r1][r2],priPr[p]);
   r1v = 1-t1;
   eta = gpd("eta",l2r[r1][r2],priPr[p]);
   g[cix] = x[gix("sc",r1,r2,p)] -
         pow(
            t1 * pow(x[gix("sa",r1,r2,p)],(eta-1)/eta)
         + r1v * pow(x[gix("sl",r1,r2,p)],(eta-1)/eta),
         eta/(eta-1)
       );
 CONSTRAIN_END_LOOP

 // mkteq17(i,p_tr).. RVAR('sl',i,p_tr)  =l=  Kt(i,p_tr);     // h16 in the presentation paper
 CONSTRAIN_START_LOOP(secPr,11)
   k = gpd("k",l2r[r1][r2],secPr[p]);
   g[cix] = x[gix("sl",r1,r2,p+nPriPr)]-k;
 CONSTRAIN_END_LOOP
00442
00443   // mkeq26(i,prd,j).. RVAR('pl',j,prd)-RVAR('pl',i,prd)-CT(i,j,prd) =l= 0;
00444   CONSTRAIN_START_LOOP(allPr,12)
00445     for (uint r2To=0;r2To<l2r[r1].size();r2To++){
00446       cix = gix(12, r1, r2, p,r2To); // attenction we must redefine it, as we are now in a r2to loop
00447       ct = gpd("ct",l2r[r1][r2],allPr[p],DATA_NOW,i2s(l2r[r1][r2To]));
00448       g[cix] = (x[gix("pl",r1,r2To,p)]-x[gix("pl",r1,r2,p)]-ct);
00449     }
00450   CONSTRAIN_END_LOOP
00451
00452   // mkteq25(i,p_tr).. sum(p_pr, a(p_pr,p_tr)*RVAR('pl',i,p_pr))+m(i,p_tr)  =g=  (RVAR('pl',i,p_tr));
        // price of raw products + transf cost > trasf product
00453   CONSTRAIN_START_LOOP(secPr,13)
00454     mv = gpd("m",l2r[r1][r2],secPr[p]);
00455     g[cix] = mv - x[gix("pl",r1,r2,p+nPriPr)];
00456     for (uint p2=0;p2<priPr.size();p2++){
00457       a = gpd("a",l2r[r1][r2],priPr[p2],DATA_NOW,secPr[p]);
00458       g[cix] += a * x[gix("pl",r1,r2,p2)];
00459     }
00460   CONSTRAIN_END_LOOP
00461
00462 //  // mkteq18(i,p_pr).. RVAR('sa',i,p_pr)+RVAR('sl',i,p_pr)  =l=  dispor(i,p_pr); // total supply lower
       than the available stock
00463 //  CONSTRAIN_START_LOOP(priPr,14)
00464 //    in = gpd("in",l2r[r1][r2],priPr[p]);
00465 //    double d1 = gix("sa",r1,r2,p);
00466 //    double d2 = gix("sl",r1,r2,p);
00467 //    g[cix] = x[gix("sa",r1,r2,p)]+x[gix("sl",r1,r2,p)]-in;
00468 //  CONSTRAIN_END_LOOP
00469
00470   // resbounds(i, p_pr_comb).. RVAR('sa',i,p_pr)+RVAR('sl',i,p_pr)  =l=  dispor(i,p_pr); // total supply
       lower than the available stock - FOR all combination subsets of ins
00471   CONSTRAIN_START_LOOP(priPrCombs,14)
00472     //ModelRegion* REG = MTHREAD->MD->getRegion(l2r[r1][r2]); // possibly slower
00473     //in = REG->inResByAnyCombination[p];
00474     in = ins[r1][r2][p];
00475     //if(p==0){
00476     //  in = 1.0; // workaround to lead -1<0 rather than 0<0 for the first (empty) subset - notneeded
00477     //}
00478     g[cix] = -in;
00479     for(uint i=0;i<priPrCombs[p].size();i++){
00480       g[cix] += x[gix("sa",r1,r2,priPrCombs[p][i])]+x[gix("sl",r1,r2,
      priPrCombs[p][i])];
00481     }
00482     g[cix] -= overharvestingAllowance; //0.02 don't work always, expecially
       intermediate scnearios, 0.1 seems to work but produce a large artefact 20160219: made it a parameter
00483
00484   CONSTRAIN_END_LOOP
00485
00486   //CONSTRAIN_START_LOOP(priPr,15)
00487   //    g[cix] = x[gix("st",r1,r2,p)]-(x[gix("sl",r1,r2,p)]+x[gix("sa",r1,r2,p)]);
00488   //CONSTRAIN_END_LOOP
00489
00490   return true;
00491 }
```

4.30.3.14 bool eval_f (Index *n*, const Number ∗ *x*, bool *new_x*, Number & *obj_value*) [virtual]

Original method from Ipopt to return the objective value remains unchanged

Definition at line 779 of file Opt.cpp.

```
00779                                                                   {
00780   eval_obj(n,x,obj_value);
00781
00782   return true;
00783 }
```

4.30.3.15 bool eval_g (Index *n*, const Number ∗ *x*, bool *new_x*, Index *m*, Number ∗ *g*) [virtual]

Original method from Ipopt to return the constraint residuals remains unchanged

Definition at line 794 of file Opt.cpp.

```
00794                                                                    {
00795
00796   eval_constraints(n,x,m,g);
00797
00798   return true;
00799 }
```

4.30.3.16 bool eval_grad_f (Index *n*, const Number ∗ *x*, bool *new_x*, Number ∗ *grad_f*) [virtual]

Original method from Ipopt to return the gradient of the objective remains unchanged

Definition at line 786 of file Opt.cpp.

```
00786                                                                     {
00787 
00788   gradient(tag_f,n,x,grad_f);
00789 
00790   return true;
00791 }
```

4.30.3.17 bool eval_h (Index *n*, const Number ∗ *x*, bool *new_x*, Number *obj_factor*, Index *m*, const Number ∗ *lambda*, bool *new_lambda*, Index *nele_hess*, Index ∗ *iRow*, Index ∗ *jCol*, Number ∗ *values*) [virtual]

Original method from Ipopt to return: 1) The structure of the hessian of the lagrangian (if "values" is NULL) 2) The values of the hessian of the lagrangian (if "values" is not NULL)remains unchanged

Definition at line 828 of file Opt.cpp.

```
00829                                                                                        {
00830 
00831 
00832   if (values == NULL) {
00833     // return the structure. This is a symmetric matrix, fill the lower left
00834     // triangle only.
00835 
00836     for(Index idx=0; idx<nnz_L; idx++)
00837       {
00838   iRow[idx] = rind_L[idx];
00839   jCol[idx] = cind_L[idx];
00840       }
00841   }
00842   else {
00843     // return the values. This is a symmetric matrix, fill the lower left
00844     // triangle only
00845 
00846     for(Index idx = 0; idx<n ; idx++)
00847       x_lam[idx] = x[idx];
00848     for(Index idx = 0; idx<m ; idx++)
00849       x_lam[n+idx] = lambda[idx];
00850     x_lam[n+m] = obj_factor;
00851 
00852     sparse_hess(tag_L, n+m+1, 1, x_lam, &nnz_L_total, &
      rind_L_total, &cind_L_total, &hessval, options_L);
00853 
00854     Index idx = 0;
00855     for(Index idx_total = 0; idx_total <nnz_L_total ; idx_total++)
00856       {
00857   if((rind_L_total[idx_total] < (unsigned int) n) && (cind_L_total[idx_total] < (
      unsigned int) n))
00858     {
00859       values[idx] = hessval[idx_total];
00860       idx++;
00861     }
00862       }
00863   }
00864 
00865   return true;
00866 
00867   //return false;
00868 }
```

4.30.3.18 bool eval_jac_g (Index *n*, const Number * *x*, bool *new_x*, Index *m*, Index *nele_jac*, Index * *iRow*, Index * *jCol*, Number * *values*) [virtual]

Original method from Ipopt to return: 1) The structure of the jacobian (if "values" is NULL) 2) The values of the jacobian (if "values" is not NULL)remains unchanged

Definition at line 802 of file Opt.cpp.

```
00803                                                           {
00804   if (values == NULL) {
00805     // return the structure of the jacobian
00806 
00807     for(Index idx=0; idx<nnz_jac; idx++)
00808       {
00809   iRow[idx] = rind_g[idx];
00810   jCol[idx] = cind_g[idx];
00811       }
00812   }
00813   else {
00814     // return the values of the jacobian of the constraints
00815 
00816     sparse_jac(tag_g, m, n, 1, x, &nnz_jac, &rind_g, &cind_g, &
      jacval, options_g);
00817 
00818     for(Index idx=0; idx<nnz_jac; idx++)
00819       {
00820   values[idx] = jacval[idx];
00821 
00822       }
00823   }
00824   return true;
00825 }
```

4.30.3.19 bool eval_obj (Index *n*, const T * *x*, T & *obj_value*)

Template to return the objective value

Define the objective function

Definition at line 220 of file Opt.cpp.

```
00220                                                {
00221 
00222   double aa, bb, dc0, sigma, a_pr, ct, m, zeromax,supCorr2;
00223   obj_value = 0.;
00224   zeromax = 0.;
00225 
00226   for (uint r1=0;r1<l2r.size();r1++){
00227     for (uint r2=0;r2<l2r[r1].size();r2++){
00228     //  // consumer's surplus..
00229     //  sum ((i,p_tr),
00230     //    AA(i,p_tr)*(RVAR('dc',i,p_tr)**((sigma(p_tr)+1)/sigma(p_tr)))
00231     //    - AA(i,p_tr)*((0.5*dc0(i,p_tr))**((sigma(p_tr)+1)/sigma(p_tr)))
00232     //    - RVAR('pc',i,p_tr)*RVAR('dc',i,p_tr)
00233     //  )
00234     // 20161003: TODO: check if subsidies should enter also the obj function other than the bounds
       equations. For the moment, as agreed with Sylvain, they are left outside the obj function, but I am not sure of it.
00235       for (uint p=0;p<secPr.size();p++){
00236         aa = gpd("aa",l2r[r1][r2],secPr[p]);
00237         sigma = gpd("sigma",l2r[r1][r2],secPr[p]);
00238         dc0 = gpd("dc",l2r[r1][r2],secPr[p],secondYear);
00239         obj_value += aa*pow(mymax(zeromax,x[gix("dc",r1,r2,p)]),(sigma+1)/sigma)-aa*pow(
      mymax(zeromax,0.5*dc0),(sigma+1)/sigma)-x[gix("pc",r1,r2,p+nPriPr)]*x[
      gix("dc",r1,r2,p)];
00240       }
00241       //  // producers surplus..
00242       //  + sum((i,p_pr),
00243       //    RVAR('pc',i,p_pr)*RVAR('sc',i,p_pr)
00244       //    - BB(i,p_pr)*(RVAR('sc',i,p_pr)**((sigma(p_pr)+1)/sigma(p_pr)))
00245       //  )
00246       for (uint p=0;p<priPr.size();p++){
00247         bb = gpd("bb",l2r[r1][r2],priPr[p]);
00248         sigma = gpd("sigmaCorr",l2r[r1][r2],priPr[p]);
00249         //supCorr2 = gpd("supCorr2",l2r[r1][r2],priPr[p]);
00250         obj_value += x[gix("pc",r1,r2,p)]*x[gix("sc",r1,r2,p)] - bb*pow(
      mymax(zeromax,x[gix("sc",r1,r2,p)]),((sigma+1)/sigma));
00251       }
00252       //  // trasformations between primary products
00253       //  + sum ((i,p_pr,p_pr2),
00254       //  +RVAR('pc',i,p_pr2)*pres(p_pr,p_pr2)*RVAR('sc',i,p_pr)
00255       //  -BB(i,p_pr2)*(pres(p_pr,p_pr2)*RVAR('sc',i,p_pr))**((sigma(p_pr2)+1)/sigma(p_pr2))
00256       //   )
00257 
00258       for (uint p1=0;p1<priPr.size();p1++){
00259         for (uint p2=0;p2<priPr.size();p2++){
00260           a_pr = gpd("a_pr",l2r[r1][r2],priPr[p1],DATA_NOW,
      priPr[p2]);
00261           bb = gpd("bb",l2r[r1][r2],priPr[p2]);
00262           sigma = gpd("sigmaCorr",l2r[r1][r2],priPr[p2]);
00263           obj_value += x[gix("pc",r1,r2,p2)]*a_pr*x[gix("sc",r1,r2,p1)]-bb*pow(
      mymax(zeromax,a_pr*x[gix("sc",r1,r2,p1)]),(sigma+1)/sigma);
00264         }
00265       }
00266       //  // surplus of transport agents..
00267       //  + sum((i,j,prd), (RVAR('pl',j,prd)-RVAR('pl',i,prd)-CT(i,j,prd))*EXP(i,j,prd))
00268       for (uint p=0;p<allPr.size();p++){
00269         for (uint r2To=0;r2To<l2r[r1].size();r2To++){
00270           ct = gpd("ct",l2r[r1][r2],allPr[p],DATA_NOW,i2s(
      l2r[r1][r2To]));
00271           obj_value += (x[gix("pl",r1,r2To,p)]-x[gix("pl",r1,r2,p)]-ct)*x[
      gix("rt",r1,r2,p,r2To)];
00272         }
00273       }
00274 
00275       //  // transformers surplus..
00276       //  + sum((i,p_tr), (RVAR('pl',i,p_tr)-m(i,p_tr))*(RVAR('sl',i,p_tr))) // attenction it's local. if
       we include w imports or p expports this have to change
00277       for (uint p=0;p<secPr.size();p++){
00278         m = gpd("m",l2r[r1][r2],secPr[p]);
00279         obj_value += (x[gix("pl",r1,r2,p+nPriPr)]-m)*x[gix("sl",r1,r2,p+
      nPriPr)];
00280       }
00281       //  - sum((i,p_pr), RVAR('pl',i,p_pr)*RVAR('dl',i,p_pr))               // to total and an other
       equation total=local+abroad should be added
00282       for (uint p=0;p<priPr.size();p++){
00283         obj_value -= x[gix("pl",r1,r2,p)]*x[gix("dl",r1,r2,p)];
00284       }
00285     } // end of each lev2 regions
00286 
00287   } //end of each r1 regions
00288 
00289   //obj_value = -obj_value; // we want maximisation, ipopt minimize! (donei n the options - scaling obj
       function)
00290 
00291  //exit(0);
00292   return true;
00293   // checked 20120802 this function is ok with gams, both in input and in output of the preoptimisation
       stage
00294 
00295 }
```

4.30.3.20 void finalize_solution (SolverReturn *status,* Index *n,* const Number ∗ *x,* const Number ∗ *z_L,* const Number ∗ *z_U,* Index *m,* const Number ∗ *g,* const Number ∗ *lambda,* Number *obj_value,* const IpoptData ∗ *ip_data,* IpoptCalculatedQuantities ∗ *ip_cq*) [virtual]

This method is called when the algorithm is complete so the TNLP can store/write the solution

Definition at line 689 of file Opt.cpp.

```
00692                                                                                                {
00693 
00694   printf("\n\nObjective value\n");
00695   printf("f(x*) = %e\n", obj_value);
00696 
00697   // --> here is where to code the assignment of optimal values to to spd()
00698 
00699   VarMap::iterator viter;
00700 
00701   // fixing the starting points for each variable at the level of the previous years
00702   for (viter = vars.begin(); viter != vars.end(); ++viter) {
00703     //string debugs = viter->first;
00704     int vdomtype = viter->second.domain;
00705     if(vdomtype==DOM_PRI_PR){
```

```
00706       for(uint r1=0;r1<l2r.size();r1++){
00707         for(uint r2=0;r2<l2r[r1].size();r2++){
00708           for(uint p=0;p<priPr.size();p++){
00709             spd(x[gix(viter->first,r1,r2,p)],viter->first,l2r[r1][r2],
      priPr[p]);
00710           }
00711         }
00712       }
00713     } else if (vdomtype==DOM_SEC_PR) {
00714       for(uint r1=0;r1<l2r.size();r1++){
00715         for(uint r2=0;r2<l2r[r1].size();r2++){
00716           for(uint p=0;p<secPr.size();p++){
00717             spd(x[gix(viter->first,r1,r2,p)],viter->first,l2r[r1][r2],
      secPr[p]);
00718
00719           }
00720         }
00721       }
00722     } else if (vdomtype==DOM_ALL_PR) {
00723       for(uint r1=0;r1<l2r.size();r1++){
00724         for(uint r2=0;r2<l2r[r1].size();r2++){
00725           for(uint p=0;p<allPr.size();p++){
00726             spd(x[gix(viter->first,r1,r2,p)],viter->first,l2r[r1][r2],
      allPr[p]);
00727           }
00728         }
00729       }
00730     } else if (vdomtype==DOM_R2_ALL_PR) {
00731       for(uint r1=0;r1<l2r.size();r1++){
00732         for(uint r2=0;r2<l2r[r1].size();r2++){
00733           for(uint p=0;p<allPr.size();p++){
00734             for(uint r2To=0;r2To<l2r[r1].size();r2To++){
00735               //if(x[gix(viter->first,r1,r2,p,r2To)] > 0){
00736               //  cout << l2r[r1][r2] << "\t" << allPr[p] << "\t" << l2r[r1][r2To] << "\t" <<
       x[gix(viter->first,r1,r2,p,r2To)] << endl;
00737               //}
00738               spd(x[gix(viter->first,r1,r2,p,r2To)],viter->first,l2r[r1][r2],
      allPr[p],DATA_NOW,false,i2s(l2r[r1][r2To]));
00739             }
00740           }
00741         }
00742       }
00743     } else {
00744       msgOut(MSG_CRITICAL_ERROR,"Try to setting the solved value of a variable of
       unknow type ("+viter->first+")");
00745     }
00746   }
00747
00748     // memory deallocation of ADOL-C variables
00749     delete[] x_lam;
00750
00751     free(rind_g);
00752     free(cind_g);
00753
00754     delete[] rind_L;
00755     delete[] cind_L;
00756
00757     free(rind_L_total);
00758     free(cind_L_total);
00759     free(jacval);
00760     free(hessval);
00761
00762     for (int i=0;i<n+m+1;i++) {
00763         free(HP_t[i]);
00764     }
00765     free(HP_t);
00766
00767 }
```

4.30.3.21 const int gdt (const string & *varName*) [protected]

Get the domain type of a given variable.

Definition at line 1248 of file Opt.cpp.

```
01248                             { // get domain type
01249   VarMap::const_iterator p;
01250   p=vars.find(varName);
01251   if(p != vars.end()) {
01252     return p->second.domain;
01253   }
01254   else {
01255     msgOut(MSG_CRITICAL_ERROR, "Asking the domain type of a variable ("+varName+")
       that doesn't exist!");
01256     return 0;
01257   }
01258 }
```

4.30.3.22 const int gdt (const int & *cn*) [protected]

Get the domain type of a given constrain.

Definition at line 1261 of file Opt.cpp.

```
01261                     { // get domain type
01262   return cons.at(cn).domain;
01263 }
```

4.30.3.23 void generate_tapes (Index *n,* Index *m,* Index & *nnz_jac_g,* Index & *nnz_h_lag*) [virtual]

Method to generate the required tapes Copied from `http://bocop.org/`

test avec "<=" (avant on avait "<" : bug, acces memoire non allouee)

valgrind : invalid read

test avec "<=" (pour etre coherent avec la remarque ci dessus, mais pas de cas test, a verifier)

test

test

Definition at line 874 of file Opt.cpp.

```
00874                                                                          {
00875     /// Copied from http://bocop.org/
00876     Number *xp     = new double[n];
00877     Number *lamp   = new double[m];
00878     Number *zl     = new double[m];
00879     Number *zu     = new double[m];
00880
00881     adouble *xa    = new adouble[n];
00882     adouble *g     = new adouble[m];
00883     adouble *lam   = new adouble[m];
00884     adouble sig;
00885     adouble obj_value;
00886
00887     double dummy;
00888 //  double *jacval;
00889
00890     int i,j,k,l,ii;
00891
00892     x_lam    = new double[n+m+1];
00893
00894 //  cout << " Avant get_start" << endl;
00895     get_starting_point(n, 1, xp, 0, zl, zu, m, 0, lamp);
00896 //   cout << " Apres get_start" << endl;
00897
00898     //if(initOpt){ // that's funny, if I use this I get it slighly longer times, whatever I then use
       trace_off() or trace_off(1) (save to disk, seems unnecessary). If I use regenerated tapes I have also slighly
       inaccurate results.
00899     trace_on(tag_f);
00900
00901     for(Index idx=0;idx<n;idx++)
00902         xa[idx] <<= xp[idx];
00903
00904     eval_obj(n,xa,obj_value);
00905
00906     obj_value >>= dummy;
```

```

    trace_off();

    trace_on(tag_g);

    for(Index idx=0;idx<n;idx++)
        xa[idx] <<= xp[idx];

    eval_constraints(n,xa,m,g);


    for(Index idx=0;idx<m;idx++)
        g[idx] >>= dummy;

    trace_off();

    trace_on(tag_L);

    for(Index idx=0;idx<n;idx++)
        xa[idx] <<= xp[idx];
    for(Index idx=0;idx<m;idx++)
        lam[idx] <<= 1.0;
    sig <<= 1.0;

    eval_obj(n,xa,obj_value);

    obj_value *= sig;
    eval_constraints(n,xa,m,g);

    for(Index idx=0;idx<m;idx++)
        obj_value += g[idx]*lam[idx];

    obj_value >>= dummy;

    trace_off();
    //} // end of if initOpt()



    rind_g = NULL;
    cind_g = NULL;

    options_g[0] = 0;          /* sparsity pattern by index domains (default) */
    options_g[1] = 0;          /*                         safe mode (default) */
    options_g[2] = -1;         /*                     &jacval is not computed */
    options_g[3] = 0;          /*                column compression (default) */

    jacval=NULL;

    sparse_jac(tag_g, m, n, 0, xp, &nnz_jac, &rind_g, &cind_g, &
jacval, options_g);

    options_g[2] = 0;
    nnz_jac_g = nnz_jac;

    unsigned int  **JP_f=NULL;                /* compressed block row storage */
    unsigned int  **JP_g=NULL;                /* compressed block row storage */
    unsigned int  **HP_f=NULL;                /* compressed block row storage */
    unsigned int  **HP_g=NULL;                /* compressed block row storage */
    unsigned int  *HP_length=NULL;            /* length of arrays */
    unsigned int  *temp=NULL;                 /* help array */

    int ctrl_H;

    JP_f = (unsigned int **) malloc(sizeof(unsigned int*));
    JP_g = (unsigned int **) malloc(m*sizeof(unsigned int*));
    HP_f = (unsigned int **) malloc(n*sizeof(unsigned int*));
    HP_g = (unsigned int **) malloc(n*sizeof(unsigned int*));
    HP_t = (unsigned int **) malloc((n+m+1)*sizeof(unsigned int*));
    HP_length = (unsigned int *) malloc((n)*sizeof(unsigned int));
    ctrl_H = 0;

    hess_pat(tag_f, n, xp, HP_f, ctrl_H);

    indopro_forward_safe(tag_f, 1, n, xp, JP_f);
    indopro_forward_safe(tag_g, m, n, xp, JP_g);
    nonl_ind_forward_safe(tag_g, m, n, xp, HP_g);

    for (i=0;i<n;i++)
    {
        if (HP_f[i][0]+HP_g[i][0]!=0)
        {
            if (HP_f[i][0]==0) // number of non zeros in the i-th row
            {
                HP_t[i] = (unsigned int *) malloc((HP_g[i][0]+HPOFF)*sizeof(unsigned int));
                for(j=0;j<=(int) HP_g[i][0];j++)
                {
                    HP_t[i][j] = HP_g[i][j];
                }
                HP_length[i] = HP_g[i][0]+HPOFF;
            }
            else
            {
                if (HP_g[i][0]==0) // number of non zeros in the i-th row
                {
                    HP_t[i] = (unsigned int *) malloc((HP_f[i][0]+HPOFF)*sizeof(unsigned int));
                    for(j=0;j<=(int) HP_f[i][0];j++)
                    {
                        HP_t[i][j] = HP_f[i][j];
                    }
                    HP_length[i] = HP_f[i][0]+HPOFF;
                }
                else
                {
                    HP_t[i] = (unsigned int *) malloc((HP_f[i][0]+HP_g[i][0]+HPOFF)*sizeof(unsigned int));
                    k = l = j = 1;
                    while ((k<=(int) HP_f[i][0]) && (l <= (int) HP_g[i][0]))
                    {
                        if (HP_f[i][k] < HP_g[i][l])
                        {
                            HP_t[i][j]=HP_f[i][k];
                            j++; k++;
                        }
                        else
                        {
                            if (HP_f[i][k] == HP_g[i][l])
                            {
                                HP_t[i][j]=HP_f[i][k];
                                l++;j++;k++;
                            }
                            else
                            {
                                HP_t[i][j]=HP_g[i][l];
                                j++;l++;
                            }
                        }
                    } // end while

                    // Fill the end of the vector if HP_g[i][0] < HP_f[i][0]
                    for(ii=k;ii<=(int) HP_f[i][0];ii++)
                    {
                        HP_t[i][j] = HP_f[i][ii];
                        j++;
                    }

                    // Fill the end of the vector if HP_f[i][0] < HP_g[i][0]
                    for(ii=l;ii<=(int) HP_g[i][0];ii++)
                    {
                        HP_t[i][j] = HP_g[i][ii];
                        j++;
                    }

                }
            }
            HP_t[i][0]=j-1; // set the first element with the number of non zeros in the i-th line
            HP_length[i] = HP_f[i][0]+HP_g[i][0]+HPOFF; // length of the i-th line
        }
        else
        {
            HP_t[i] = (unsigned int *) malloc((HPOFF+1)*sizeof(unsigned int));
            HP_t[i][0]=0;
            HP_length[i]=HPOFF;
        }

//     if (i==(int)n-1)
//     {
//       cout << " DISPLAY FINAL TIME HP : " << endl;
//       for (ii=0;ii<=(int)HP_length[i];ii++)
//         cout << " -------> HP[last][" << ii << "] = " << HP_t[i][ii] << endl;
//     }
    }

//   cout << " Avant les boucles" << endl;
//   cout << " m = " << m << endl;

    for (i=0;i<m;i++)
    {
//     cout << i << " --> nnz JP_g = " << JP_g[i][0]+1 << " -- ";
        HP_t[n+i] = (unsigned int *) malloc((JP_g[i][0]+1)*sizeof(unsigned int));
        HP_t[n+i][0]=JP_g[i][0];

//     cout << HP_t[n+i][0] << endl;

```

```
        for(j=1;j<= (int) JP_g[i][0];j++)
        {
            HP_t[n+i][j]=JP_g[i][j];
//       cout << " ---------> " << HP_t[n+i][j] << endl;
//       cout << " --> HP_length[" << JP_g[i][j] << "] = " << HP_length[JP_g[i][j]] << "  --  HP_t[" << JP_g[i][j] << "][0] = " << HP_t[JP_g[i][j]][0]+1 << endl;
            // We write the rows allocated in the previous "for" loop
            // If the memory allocated for the row is not big enough :
            if (HP_length[JP_g[i][j]] <= HP_t[JP_g[i][j]][0]+1) //! test avec "<=" (avant on avait "<" : bug, acces memoire non allouee)
            {
//         cout << " ---------> WARNING " << endl;
//         cout << " At index " << JP_g[i][j] << endl;


                // save a copy of existing vector elements :
                temp = (unsigned int *) malloc((HP_t[JP_g[i][j]][0]+1)*sizeof(unsigned int));
                for(l=0;l<=(int)HP_t[JP_g[i][j]][0];l++)
                {
                    temp[l] = HP_t[JP_g[i][j]][l]; //! valgrind : invalid read
//           cout << " -------> l = " << l << " -- " << temp[l] << endl;
                }

//         cout << " ----------->  DISPLAY " << endl;
//         for(l=0;l<=(int)HP_t[JP_g[i][j]][0];l++)
//         {
//           temp[l] = HP_t[JP_g[i][j]][l]; //! valgrind : invalid read & write
//           cout << " -------> HP[machin][" << l << "] = " << HP_t[JP_g[i][j]][l] << endl; //! valgrind : invalid read
//         }


                // Free existing row, and allocate more memory for it :
//         cout << " Avant free  --> pointeur = " <<HP_t[JP_g[i][j]]<< endl;
                unsigned int machin = JP_g[i][j];
                free(HP_t[machin]); // !Problem double free or corruption
//         cout << " Apres free --> pointeur = " <<HP_t[JP_g[i][j]]<< endl;

                HP_t[JP_g[i][j]] = (unsigned int *) malloc(2*HP_length[JP_g[i][j]]*sizeof(unsigned int));
                HP_length[JP_g[i][j]] = 2*HP_length[JP_g[i][j]];

                // Put back the values in this bigger vector :
                for(l=0;l<=(int)temp[0];l++)
                    HP_t[JP_g[i][j]][l] =temp[l];
                free(temp);

//         HP_t[JP_g[i][j]] = (unsigned int*) realloc (HP_t[JP_g[i][j]], 2*HP_length[JP_g[i][j]] * sizeof(unsigned int));
//         HP_length[JP_g[i][j]] = 2*HP_length[JP_g[i][j]];
            }
            HP_t[JP_g[i][j]][0] = HP_t[JP_g[i][j]][0]+1; // The size of the row is one greater than before
            HP_t[JP_g[i][j]][HP_t[JP_g[i][j]][0]] = i+n; // Now adding the element at the end //! valgrind : invalid write
        }
    }
//   cout << " Apres les boucles" << endl;

    for(j=1;j<= (int) JP_f[0][0];j++)
    {
        if (HP_length[JP_f[0][j]] <= HP_t[JP_f[0][j]][0]+1) //! test avec "<=" (pour etre coherent avec la remarque ci dessus, mais pas de cas test, a verifier)
        {
            temp = (unsigned int *) malloc((HP_t[JP_f[0][j]][0])*sizeof(unsigned int));
            for(l=0;l<=(int)HP_t[JP_f[0][j]][0];l++)
                temp[l] = HP_t[JP_f[0][j]][l];
            free(HP_t[JP_f[0][j]]);
            HP_t[JP_f[0][j]] = (unsigned int *) malloc(2*HP_length[JP_f[0][j]]*sizeof(unsigned int));
            HP_length[JP_f[0][j]] = 2*HP_length[JP_f[0][j]];
            for(l=0;l<=(int)temp[0];l++)
                HP_t[JP_f[0][j]][l] =temp[l];
            free(temp);
        }
        HP_t[JP_f[0][j]][0] = HP_t[JP_f[0][j]][0]+1;
        HP_t[JP_f[0][j]][HP_t[JP_f[0][j]][0]] = n+m;
    }

    HP_t[n+m] = (unsigned int *) malloc((JP_f[0][0]+2)*sizeof(unsigned int));
    HP_t[n+m][0]=JP_f[0][0]+1;
    for(j=1;j<= (int) JP_f[0][0];j++)
        HP_t[n+m][j]=JP_f[0][j];
    HP_t[n+m][JP_f[0][0]+1]=n+m;

    set_HP(tag_L,n+m+1,HP_t); // set sparsity pattern for the Hessian

    nnz_h_lag = 0;
    for (i=0;i<n;i++)
    {
        for (j=1;j<=(int) HP_t[i][0];j++)
            if ((int) HP_t[i][j] <= i)
                nnz_h_lag++;
        free(HP_f[i]);
        free(HP_g[i]);
    }
    nnz_L = nnz_h_lag;

    options_L[0] = 0;
    options_L[1] = 1;

    rind_L_total = NULL;
    cind_L_total = NULL;
    hessval = NULL;

    sparse_hess(tag_L, n+m+1, -1, xp, &nnz_L_total, &rind_L_total, &cind_L_total, &hessval, options_L);

    rind_L = new unsigned int[nnz_L];
    cind_L = new unsigned int[nnz_L];
    rind_L_total = (unsigned int*) malloc(nnz_L_total*sizeof(unsigned int)); //! test
    cind_L_total = (unsigned int*) malloc(nnz_L_total*sizeof(unsigned int)); //! test

    unsigned int ind = 0;

    for (int i=0;i<n;i++)
        for (unsigned int j=1;j<=HP_t[i][0];j++)
    {
        if (((int) HP_t[i][j]>=i) &&((int) HP_t[i][j]<n))
        {
            rind_L[ind] = i;
            cind_L[ind++] = HP_t[i][j];
        }
    }

    ind = 0;
    for (int i=0;i<n+m+1;i++)
        for (unsigned int j=1;j<=HP_t[i][0];j++)
    {
        if ((int) HP_t[i][j]>=i)
        {
            rind_L_total[ind] = i;
            cind_L_total[ind++] = HP_t[i][j];
        }
    }

    for (i=0;i<m;i++) {
        free(JP_g[i]);
    }

    free(JP_f[0]);
    free(JP_f);
    free(JP_g);
    free(HP_f);
    free(HP_g);
    free(HP_length);

    delete[] lam;
    delete[] g;
    delete[] xa;
    delete[] zu;
    delete[] zl;
    delete[] lamp;
    delete[] xp;

}
```

4.30.3.24 bool get_bounds_info (Index *n*, Number ∗ *x_l*, Number ∗ *x_u*, Index *m*, Number ∗ *g_l*, Number ∗ *g_u*) [virtual]

Method to return the bounds for my problem

Definition at line 591 of file Opt.cpp.

```
00591                                                                                     {
00592
00593   // Set the bounds for the endogenous variables..
00594   for (Index i=0; i<n; i++) {
00595     x_l[i] =  getBoundByIndex(LBOUND,i);
00596     x_u[i] =  getBoundByIndex(UBOUND,i);
00597   }
00598
00599   // Set the bounds for the constraints..
00600   for (Index i=0; i<m; i++) {
00601     int direction = getConstrainDirectionByIndex(i);
00602     switch (direction){
00603       case CONSTR_EQ:
00604         g_l[i] = 0.;
00605         g_u[i] = 0.;
00606         break;
00607       case CONSTR_LE0:
00608         g_l[i] = -2e19;
00609         g_u[i] = 0.;
00610         break;
00611       case CONSTR_GE0:
00612         g_l[i] = 0.;
00613         g_u[i] = 2e19;
00614         break;
00615     }
00616   }
00617   return true;
00618 }
```

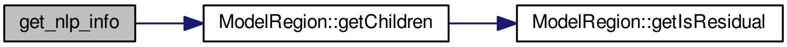
4.30.3.25 bool get_nlp_info (Index & *n*, Index & *m*, Index & *nnz_jac_g*, Index & *nnz_h_lag*, IndexStyleEnum & *index_style*) [virtual]

Method to return some info about the nlp

Definition at line 510 of file Opt.cpp.

```
00511                                                                 {
00512
00513
00514   if(initOpt){
00515     // does this initialisation code only once
00516     priPr = MTHREAD->MD->getStringVectorSetting("priProducts");
00517     secPr = MTHREAD->MD->getStringVectorSetting("secProducts");
00518     allPr = priPr;
00519     allPr.insert( allPr.end(), secPr.begin(), secPr.end() );
00520     nPriPr = priPr.size();
00521     nSecPr = secPr.size();
00522     nAllPr = allPr.size();
00523     std::vector<int> l1regIds =  MTHREAD->MD->getRegionIds(1, true);
00524     nL2r =  MTHREAD->MD->getRegionIds(2, true).size();
00525     firstYear = MTHREAD->MD->getIntSetting("initialYear");
00526     secondYear = firstYear+1;
00527     worldCodeLev2 = MTHREAD->MD->getIntSetting("worldCodeLev2");
00528
00529     for(uint i=0;i<l1regIds.size();i++){
00530       std::vector<int> l2ChildrenIds;
00531       ModelRegion* l1Region = MTHREAD->MD->getRegion(l1regIds[i]);
00532       std::vector<ModelRegion*> l2Childrens = l1Region->getChildren(true);
00533       for(uint j=0;j<l2Childrens.size();j++){
00534         l2ChildrenIds.push_back(l2Childrens[j]->getRegId());
00535       }
00536       if(l2ChildrenIds.size()){
00537         l2r.push_back(l2ChildrenIds);
00538       }
00539     }
00540
00541     // Create a vector with all possible combinations of primary products
00542     priPrCombs = MTHREAD->MD->createCombinationsVector(
      nPriPr);
00543     nPriPrCombs = priPrCombs.size();
00544
00545     // put the variables and their domain in the vars map
00546     declareVariables();
00547
00548     // declaring the contrains...
00549     declareConstrains();
00550
00551     // calculate number of variables and constrains..
00552     calculateNumberVariablesConstrains();
00553
00554     // cache initial positions (variables and constrains)..
00555     cacheInitialPosition();
00556
00557     // cache initial positions (variables and constrains)..
00558     cachePositions();
00559
00560     //tempDebug();
00561
00562     //debugPrintParameters();
00563
00564   } // finish initialisation things to be done only the first year
00565
00566   previousYear = MTHREAD->SCD->getYear()-1; // this has to be done EVERY years !!
00567
00568   n = nVar; // 300; // nVar;
00569   m = nCons; // 70; // nCons;
00570
00571   overharvestingAllowance = MTHREAD->MD->getDoubleSetting("overharvestingAllowance",DATA_NOW);
00572
00573   copyInventoryResourses();
00574
00575   generate_tapes(n, m, nnz_jac_g, nnz_h_lag);
00576
00577   //if(initOpt){
00578   //  calculateSparsityPatternJ();
00579   //  calculateSparsityPatternH();
00580     //tempDebug();
00581   //}
00582
00583   // use the C style indexing (0-based)
00584   index_style = C_STYLE;
00585
00586   initOpt=false;
00587   return true;
00588 }
```

Here is the call graph for this function:



4.30.3.26 bool get_starting_point (Index *n*, bool *init_x*, Number ∗ *x*, bool *init_z*, Number ∗ *z_L*, Number ∗ *z_U*, Index *m*, bool *init_lambda*, Number ∗ *lambda*) [virtual]

Method to return the starting point for the algorithm

Definition at line 621 of file Opt.cpp.

```
00622                                                                  {
00623
00624   // function checked on 20120724 on a subset of 3 regions and 4 products. All variables initial values are correctly those outputed by gams in 2006.
00625   //int thisYear = MTHREAD->SCD->getYear();
00626   //int initialOptYear = MTHREAD->MD->getIntSetting("initialOptYear");
00627   //if(thisYear != initialOptYear) return true;
00628
00629   //msgOut(MSG_DEBUG,"Giving optimising variables previous years value as starting point");
00630   // Here, we assume we only have starting values for x, if you code
00631   // your own NLP, you can provide starting values for the others if
00632   // you wish.
00633   assert(init_x == true);
00634   assert(init_z == false);
00635   assert(init_lambda == false);
00636
00637   VarMap::iterator viter;
00638
00639   // fixing the starting points for each variable at the level of the previous years
00640   for (viter = vars.begin(); viter != vars.end(); ++viter) {
00641     //string debugs = viter->first;
00642     int vdomtype = viter->second.domain;
00643     if(vdomtype==DOM_PRI_PR){
00644       for(uint r1=0;r1<l2r.size();r1++){
00645         for(uint r2=0;r2<l2r[r1].size();r2++){
00646           for(uint p=0;p<priPr.size();p++){
00647             x[gix(viter->first,r1,r2,p)]= gpd(viter->first,l2r[r1][r2],
      priPr[p],previousYear);
00648           }
00649         }
00650       }
00651     } else if (vdomtype==DOM_SEC_PR) {
00652       for(uint r1=0;r1<l2r.size();r1++){
00653         for(uint r2=0;r2<l2r[r1].size();r2++){
00654           for(uint p=0;p<secPr.size();p++){
00655             x[gix(viter->first,r1,r2,p)]= gpd(viter->first,l2r[r1][r2],
      secPr[p],previousYear);
00656           }
00657         }
00658       }
00659     } else if (vdomtype==DOM_ALL_PR) {
00660       for(uint r1=0;r1<l2r.size();r1++){
00661         for(uint r2=0;r2<l2r[r1].size();r2++){
00662           for(uint p=0;p<allPr.size();p++){
00663             x[gix(viter->first,r1,r2,p)]= gpd(viter->first,l2r[r1][r2],
      allPr[p],previousYear);
00664           }
00665         }
00666       }
00667     } else if (vdomtype==DOM_R2_ALL_PR) {
00668       for(uint r1=0;r1<l2r.size();r1++){
00669         for(uint r2=0;r2<l2r[r1].size();r2++){
00670           for(uint p=0;p<allPr.size();p++){
00671             for(uint r2To=0;r2To<l2r[r1].size();r2To++){
00672               x[gix(viter->first,r1,r2,p,r2To)]= gpd(viter->first,l2r[r1][r2],
      allPr[p],previousYear,i2s(l2r[r1][r2To]));
00673             }
00674           }
00675         }
00676       }
00677     } else {
00678       msgOut(MSG_CRITICAL_ERROR,"Try to setting the initial value of a variable of
       unknow type ("+viter->first+")");
00679     }
00680   }
00681 
00682   //msgOut(MSG_DEBUG,"Finisced initial value assignments");
00683 
00684   return true;
00685 }
```

4.30.3.27 double getBoundByIndex (const int & *bound_type,* const int & *idx*) [protected]

Return the bound of a given variable (by index)

Definition at line 1494 of file Opt.cpp.

```
01494                                                             {
01495 
01496   map <int, string>::const_iterator p;
01497   p=initPos_rev.upper_bound(idx);
01498   p--;
01499   VarMap::const_iterator p2;
01500   p2=vars.find(p->second);
01501   if(p2 != vars.end()) {
01502     if (bound_type==LBOUND){
01503       if (p2->second.l_bound_var == ""){ // this var don't specific a variable as bound
01504         return p2->second.l_bound;
01505       } else {
01506         return getDetailedBoundByVarAndIndex(p2->second,idx,
      LBOUND);
01507       }
01508     } else if (bound_type==UBOUND){
01509       if (p2->second.u_bound_var == ""){ // this var don't specific a variable as bound
01510         return p2->second.u_bound;
01511       } else {
01512         return getDetailedBoundByVarAndIndex(p2->second,idx,
      UBOUND);
01513       }
01514     } else {
01515       msgOut(MSG_CRITICAL_ERROR, "Asking the bound with a type ("+
      i2s(bound_type)+") that I don't know how to handle !");
01516     }
01517   }
01518   else {
01519     msgOut(MSG_CRITICAL_ERROR, "Asking the bound from a variable ("+p->second+")
       that doesn't exist!");
01520   }
01521   return 0.;
01522 }
```

4.30.3.28 int getConNumber (**constrain** ∗ *con*) [protected]

Return the position in the cons vector.

Definition at line 1622 of file Opt.cpp.

```
01622                                {
01623   for(uint i=0;i<cons.size();i++){
01624     if(   cons[i].name       == con->name
01625        && cons[i].comment    == con->comment
01626        && cons[i].domain     == con->domain
01627        && cons[i].direction  == con->direction){
01628       return i;
01629        }
01630   }
01631   msgOut(MSG_CRITICAL_ERROR,"Constrain didn't found in list.");
01632 }
```

4.30.3.29 **constrain** ∗ getConstrainByIndex (int *idx*) [protected]

Definition at line 1548 of file Opt.cpp.

```
01548                                {
01549   for(uint i=0;i<cons.size();i++){
01550     if(i!=cons.size()-1){
01551       if (idx >= gip(i) && idx < gip(i+1)){
01552         return &cons[i];
01553       }
01554     } else {
01555       if (idx >= gip(i) && idx < nCons){
01556       return &cons[i];
01557       }
01558     }
01559   }
01560   msgOut(MSG_CRITICAL_ERROR, "Asking contrain direction for an out of range
       contrain index!");
01561 }
```

4.30.3.30 int getConstrainDirectionByIndex (int *idx*) [protected]

Return the direction of a given constrain.

Definition at line 1478 of file Opt.cpp.

```
01478                                         {
01479   for(uint i=0;i<cons.size();i++){
01480     if(i!=cons.size()-1){
01481       if (idx >= gip(i) && idx < gip(i+1)){
01482         return cons[i].direction;
01483       }
01484     } else {
01485       if (idx >= gip(i) && idx < nCons){
01486       return cons[i].direction;
01487       }
01488     }
01489   }
01490   msgOut(MSG_CRITICAL_ERROR, "Asking contrain direction for an out of range
       contrain index!");
01491 }
```

4.30.3.31 double getDetailedBoundByVarAndIndex (const endvar & *var,* const int & *idx,* const int & *bType*) [protected]

Return the bound of a given variable given the variable and the required index. Called by getBoundByIndex().

Definition at line 1525 of file Opt.cpp.

```
01525                                                                                         {
01526   // Tested 2015.01.08 with DOM_ALL_PR, DOM_PRI_PR, DOM_ALL_PR, DOM_R2_ALL_PR.
01527   int r1,r2,p,r2to;
01528   unpack(idx,var.domain,gip(var.name),r1,r2,p,r2to,true);
01529   //cout << "getBoundByVarAndIndex():\t" << var.name << ’\t’ << idx << ’\t’ << gip(var.name) << ’\t’ << r1
       << ’\t’ << r2 << ’\t’ << p << ’\t’ << r2to << endl;
01530   //cout << "  --variables:\t" << var.l_bound_var << ’\t’ << var.u_bound_var << ’\t’ << "" << ’\t’ <<
       l2r[r1][r2] << ’\t’ << "" << ’\t’ << allPr[p] << ’\t’ << l2r[r1][r2to] << endl;
01531   if(bType==LBOUND){
01532     if(r2to){
01533        return gpd(var.l_bound_var,l2r[r1][r2],allPr[p],
      DATA_NOW,i2s(l2r[r1][r2to]));
01534     } else {
01535       return gpd(var.l_bound_var,l2r[r1][r2],allPr[p],
      DATA_NOW,i2s(l2r[r1][r2to]));
01536     }
01537   } else {
01538     if(r2to){
01539        return gpd(var.u_bound_var,l2r[r1][r2],allPr[p]);
01540     } else {
01541       //cout << gpd(var.u_bound_var,l2r[r1][r2],allPr[p]) << endl;
01542       return gpd(var.u_bound_var,l2r[r1][r2],allPr[p]);
01543     }
01544   }
01545 }
```

4.30.3.32 int getDomainElements (int *domain*)

return the number of elements of a domain

Definition at line 1444 of file Opt.cpp.

```
01444                                 {
01445   int elements = 0;
01446   switch (domain){
01447     case DOM_PRI_PR:
01448       return nL2r*nPriPr;
01449     case DOM_SEC_PR:
01450       return nL2r*nSecPr;
01451     case DOM_ALL_PR:
01452       return nL2r*nAllPr;
01453     case DOM_R2_PRI_PR:
01454       for(uint r1=0;r1<l2r.size();r1++){
01455           elements += l2r[r1].size()*l2r[r1].size()*nPriPr; // EXP(i,j,p_pr)
01456       }
01457       return elements;
01458     case DOM_R2_SEC_PR:
01459       for(uint r1=0;r1<l2r.size();r1++){
01460           elements += l2r[r1].size()*l2r[r1].size()*nSecPr; // EXP(i,j,p_tr)
01461       }
01462       return elements;
01463     case DOM_R2_ALL_PR:
01464       for(uint r1=0;r1<l2r.size();r1++){
01465           elements += l2r[r1].size()*l2r[r1].size()*nAllPr; // EXP(i,j,prd)
01466       }
01467       return elements;
01468     case DOM_SCALAR:
01469       return 1;
01470     case DOM_PRI_PR_ALLCOMBS:
01471       return nL2r*nPriPrCombs;
01472   default:
01473     msgOut(MSG_CRITICAL_ERROR, "Asking for an unknown domain type ("+
      i2s(domain)+")");
01474   }
01475 }
```

4.30.3.33 int getVarInstances (const string & *varName*)

build the matrix of the positions for a given variable or contrain

return the number of instances of a variable, given his domain type

Definition at line 1724 of file Opt.cpp.

```
01724                                            {
01725   return getDomainElements(gdt(varName));
01726 }
```

4.30.3.34 const double gfd (const string & *type_h,* const int & *regId_h,* const string & *forType_h,* const string & *diamClass_h,* const int & *year* = DATA_NOW) const [inline], [protected]

Definition at line 169 of file Opt.h.

```
00169 {return MTHREAD->MD->getForData(type_h, regId_h, forType_h, diamClass_h, year);};
```

4.30.3.35 const int gip (const string & *varName*) const [protected]

Get the initial index position of a given variable in the concatenated array.

Definition at line 1230 of file Opt.cpp.

```
01230                                   { // get initial position
01231   map<string, int>::const_iterator p;
01232   p=initPos.find(varName);
01233   if(p != initPos.end()) {
01234     return p->second;
01235   }
01236   else {
01237     msgOut(MSG_CRITICAL_ERROR, "Asking the initial position in the concatenated
       array of a variable ("+varName+") that doesn't exist!");
01238     return 0;
01239   }
01240 }
```

4.30.3.36 const int gip (const int & *cn*) const [protected]

Return the initial index position of a certain constrain.

Definition at line 1243 of file Opt.cpp.

```
01243                            { // get initial position
01244   return cInitPos.at(cn);
01245 }
```

4.30.3.37 const int gix (const string & *varName*, const int & *r1Ix*, const int & *r2Ix*, const int & *prIx*, const int & *r2IxTo* = 0) const [protected]

Get the index in the concatenated array gived a certain var name, the reg lev1 index, the reg lev2 index and the prod. index.

Definition at line 1307 of file Opt.cpp.

```
01307                                                                                                                        {
01308   // attenction, for computational reasons we are not checking the call is within vectors limits!!!
01309   map <string, vector < vector < vector < vector <int> > > > >::const_iterator p;
01310   p=vpositions.find(varName);
01311   if(p != vpositions.end()) {
01312     return p->second[r1Ix][r2Ix][prIx][r2IxTo];
01313   }
01314   else {
01315     msgOut(MSG_CRITICAL_ERROR, "Asking the position of a variable ("+varName+")
      that doesn’t exist!");
01316     return 0;
01317   }
01318 }
```

4.30.3.38 const int gix (const int & *cn*, const int & *r1Ix*, const int & *r2Ix*, const int & *prIx*, const int & *r2IxTo* = 0) const [protected]

Get the index in the concatenated array gived a certain constrain, the reg lev1 index, the reg lev2 index and the prod. index.

Definition at line 1321 of file Opt.cpp.

```
01321                                                                                                                {
01322   return cpositions[cn][r1Ix][r2Ix][prIx][r2IxTo];
01323 }
```

4.30.3.39 const int gix_uncached (const T & *v_or_c*, int *r1Ix*, int *r2Ix*, int *prIx*, int *r2IxTo* = 0) [protected]

Get the index in the concatenated array gived a certain var name (string) or constrain index (int), the reg lev1 index, the reg lev2 index and the prod. index.

Definition at line 1266 of file Opt.cpp.

```
01266                                                                                      {
01267
01268   // attenction, for computational reason we are not checking the call is within vectors limits!!!
01269
01270   int dType = gdt(v_or_c);
01271   int othCountriesRegions = 0;
01272   int othCountriesRegions_r2case = 0;
01273   for (uint i=0;i<r1Ix;i++){
01274     othCountriesRegions += l2r[i].size();
01275   }
01276   for (uint i=0;i<r1Ix;i++){
01277     othCountriesRegions_r2case +=l2r[i].size()*l2r[i].size();
01278   }
01279
01280   switch (dType){
01281     case DOM_PRI_PR:
01282       return gip(v_or_c)+(othCountriesRegions+r2Ix)*nPriPr+prIx;
01283     case DOM_SEC_PR:
01284       return gip(v_or_c)+(othCountriesRegions+r2Ix)*nSecPr+prIx;;
01285     case DOM_ALL_PR:
01286       return gip(v_or_c)+(othCountriesRegions+r2Ix)*nAllPr+prIx;
01287     case DOM_R2_PRI_PR:
01288       return gip(v_or_c)+(othCountriesRegions_r2case)*nAllPr+(r2Ix*
      nPriPr+prIx)*l2r[r1Ix].size()+r2IxTo;
01289     case DOM_R2_SEC_PR:
01290       return gip(v_or_c)+(othCountriesRegions_r2case)*nAllPr+(r2Ix*
      nSecPr+prIx)*l2r[r1Ix].size()+r2IxTo;
01291     case DOM_R2_ALL_PR:
01292       return gip(v_or_c)+(othCountriesRegions_r2case)*nAllPr+(r2Ix*
      nAllPr+prIx)*l2r[r1Ix].size()+r2IxTo; // new 20120814, looping r1,r2,p,r2to
01293       // initial position + (other countries region pairs + same country other regions from pair + regions
       to)* number of all products+product
01294       //return gip(v_or_c)+(othCountriesRegions_r2case+r2Ix*l2r[r1Ix].size()+r2IxTo)*nAllPr+prIx; //
       looping r1,r2,r2to,p
01295     case DOM_SCALAR:
01296       return gip(v_or_c);
01297     case DOM_PRI_PR_ALLCOMBS:
01298       return gip(v_or_c)+(othCountriesRegions+r2Ix)*nPriPrCombs+prIx;
01299   default:
01300     msgOut(MSG_CRITICAL_ERROR,"Try to calculate the position of a variable (or
       constrain) of unknow type.");
01301     return 0;
01302   }
01303 }
```

4.30.3.40 const double gpd (const string & *type_h*, const int & *regId_h*, const string & *prodId_h*, const int & *year* = DATA_NOW, const string & *freeDim_h* = "") const [inline], [protected]

Definition at line 168 of file Opt.h.

```
00168 {return MTHREAD->MD->getProdData(type_h, regId_h, prodId_h, year, freeDim_h);};
```

4.30.3.41 bool intermediate_callback (AlgorithmMode *mode*, Index *iter*, Number *obj_value*, Number *inf_pr*, Number *inf_du*, Number *mu*, Number *d_norm*, Number *regularization_size*, Number *alpha_du*, Number *alpha_pr*, Index *ls_trials*, const IpoptData ∗ *ip_data*, IpoptCalculatedQuantities ∗ *ip_cq*) [virtual]

Return information on each iteration

Definition at line 1715 of file Opt.cpp.

```
01715

                                                         {
01716     int itnumber = iter;
01717     if(itnumber%10==0){
01718         msgOut(MSG_DEBUG,"Running ("+i2s(itnumber)+" iter) ..");
01719     }
01720     return true;
01721 }
```

4.30.3.42 const Number & mymax (const Number & *a*, const Number & *b*)

Definition at line 1705 of file Opt.cpp.

```
01705                                            {
01706   return (a<b)?b:a;
01707 }
```

4.30.3.43 const adouble & mymax (const adouble & *a*, const adouble & *b*)

Definition at line 1709 of file Opt.cpp.

```
01709                                              {
01710   return (a<b)?b:a;
01711 }
```

4.30.3.44 Opt& operator= (const Opt &) [protected]

4.30.3.45 void sfd (const double & *value_h*, const string & *type_h*, const int & *regId_h*, const string & *forType_h*, const string & *diamClass_h*, const int & *year* = DATA_NOW, const bool & *allowCreate* = false) const [inline], [protected]

Definition at line 171 of file Opt.h.

```
00171 {MTHREAD->MD->setForData(value_h, type_h, regId_h, forType_h, diamClass_h, year,
      allowCreate);};
```

4.30.3.46 void spd (const double & *value_h*, const string & *type_h*, const int & *regId_h*, const string & *prodId_h*, const int & *year* = DATA_NOW, const bool & *allowCreate* = false, const string & *freeDim_h* = "") const [inline], [protected]

Definition at line 170 of file Opt.h.

```
00170 {MTHREAD->MD->setProdData(value_h, type_h, regId_h, prodId_h, year, allowCreate,
      freeDim_h);};
```

4.30.3.47 void tempDebug ( ) [protected]

Definition at line 1689 of file Opt.cpp.

```
01689               {
01690 
01691   cout << "Num of variables: " <<  nVar << " - Num of constrains:" << nCons << endl;
01692   cout << "IDX;ROW;COL" << endl;
01693   for(uint i=0;i<nzhelements.size();i++){
01694     cout << i << ";" << nzhelements[i][0] << ";" << nzhelements[i][1] << endl;
01695   }
01696 
01697   cout << "Dense jacobian: " << nCons * nVar << " elements" << endl;
01698   cout << "Dense hessian:  " << nVar*(nVar-1)/2+nVar << " elements" << endl;
01699   //exit(0);
01700 
01701 }
```

4.30.3.48 void unpack (int *ix_h*, int *domain*, int *initial*, int & *r1_h*, int & *r2_h*, int & *p_h*, int & *r2to_h*, bool *fullp* = false) [protected]

Return the dimensions given a certain index, domain type and initial position.

Definition at line 1565 of file Opt.cpp.

```
01565                                                                                                     {
01566   ix_h = ix_h-initial;
01567   double ix=0;
01568   bool r2flag = false;
01569   int pIndexToAdd = 0;
01570   int np=0;
01571   if(domain==DOM_PRI_PR || domain==DOM_R2_PRI_PR) {
01572     np = nPriPr;
01573   } else if (domain==DOM_SEC_PR || domain==DOM_R2_SEC_PR) {
01574     np = nSecPr;
01575   } else if (domain==DOM_ALL_PR || domain==DOM_R2_ALL_PR) {
01576     np = nAllPr;
01577   } else if (domain=DOM_SCALAR){
01578     r1_h=0;r2_h=0;p_h=0;r2to_h=0;
01579     return;
01580   } else {
01581     msgOut(MSG_CRITICAL_ERROR,"unknow domain ("+i2s(domain)+") in unpack()
 function.");
  }
  if(domain==DOM_R2_PRI_PR || domain==DOM_R2_SEC_PR ||domain==
DOM_R2_ALL_PR){
    r2flag = true;
  }
  if(fullp && (domain==DOM_SEC_PR || domain==DOM_R2_SEC_PR)){ // changed 20140107
 (any how, previously the unpack() function was not used!!)
    pIndexToAdd = nPriPr;
    //cout << "pindexToAdd: " << pIndexToAdd << endl;
  }

  for (uint r1=0;r1<l2r.size();r1++){
    for (uint r2=0;r2<l2r[r1].size();r2++){
      for (uint p=0;p<np;p++){
        if(!r2flag){
          if(ix==ix_h){
            r1_h=r1;
            r2_h=r2;
            p_h=p+pIndexToAdd;
            r2to_h=0;
            return;
          }
          ix++;
        } else {
          for (uint r2To=0;r2To<l2r[r1].size();r2To++){
            if(ix==ix_h){
              r1_h=r1;
              r2_h=r2;
              p_h=p+pIndexToAdd;
              r2to_h=r2To;
              return;
            }
            ix++;
          }
        }
      }
    }
  }
  msgOut(MSG_CRITICAL_ERROR, "Error in unpack() function. Ix ("+
i2s(ix_h)+") can not be unpacked");
}
```

4.30.4 Member Data Documentation

4.30.4.1 vector< string > allPr [protected]

Definition at line 195 of file Opt.h.

4.30.4.2 unsigned int* cind_g [protected]

Definition at line 251 of file Opt.h.

4.30.4.3 unsigned int* cind_L [protected]

Definition at line 254 of file Opt.h.

4.30.4.4 unsigned int* cind_L_total [protected]

Definition at line 256 of file Opt.h.

4.30.4.5 vector< int > cInitPos [protected]

A vector that returns the initial index position in the concatenated array for each constrain.

Definition at line 201 of file Opt.h.

4.30.4.6 vector< constrain > cons [protected]

Definition at line 223 of file Opt.h.

4.30.4.7 vector< vector < vector < vector < vector <int> > > > > cpositions [protected]

cached position in the concatenated vector for each variables. Dimensions are contrain number, l1reg, l2reg, prod, (l2To region).

Definition at line 204 of file Opt.h.

4.30.4.8 bool debugRunOnce [protected]

Definition at line 219 of file Opt.h.

4.30.4.9 int firstYear [protected]

Definition at line 216 of file Opt.h.

4.30.4.10 double* hessval [protected]

Definition at line 257 of file Opt.h.

4.30.4.11 unsigned int** HP_t [protected]

Definition at line 249 of file Opt.h.

4.30.4.12 bool initOpt [protected]

Definition at line 222 of file Opt.h.

4.30.4.13 map<string, int> initPos [protected]

A map that returns the initial index position in the concatenated array for each variable.

Definition at line 199 of file Opt.h.

4.30.4.14 map<int, string> initPos_rev [protected]

A map with the name of the variable keyed by its initial position in the index.

Definition at line 200 of file Opt.h.

4.30.4.15 vector< vector < vector <double> > > ins [protected]

A copy of the inventoried resourses by region and primary product combination. It works also with dynamic loading of the region and the in, but it may be slower.

Definition at line 198 of file Opt.h.

4.30.4.16 double∗ jacval [protected]

Definition at line 252 of file Opt.h.

4.30.4.17 vector< vector <int> > l2r [protected]

Definition at line 196 of file Opt.h.

4.30.4.18 int nAllPr [protected]

Definition at line 208 of file Opt.h.

4.30.4.19 int nCons [protected]

Definition at line 211 of file Opt.h.

4.30.4.20 int nEqualityConstrains [protected]

Definition at line 212 of file Opt.h.

4.30.4.21 int nGreaterEqualZeroConstrains [protected]

Definition at line 214 of file Opt.h.

4.30.4.22 int nL2r [protected]

Definition at line 209 of file Opt.h.

4.30.4.23 int nLowerEqualZeroConstrains [protected]

Definition at line 213 of file Opt.h.

4.30.4.24 int nnz_jac [protected]

Definition at line 258 of file Opt.h.

4.30.4.25 int nnz_L [protected]

Definition at line 259 of file Opt.h.

4.30.4.26 int nnz_L_total [protected]

Definition at line 259 of file Opt.h.

4.30.4.27 int nPriPr [protected]

Definition at line 205 of file Opt.h.

4.30.4.28 int nPriPrCombs [protected]

Definition at line 206 of file Opt.h.

4.30.4.29 int nSecPr [protected]

Definition at line 207 of file Opt.h.

4.30.4.30 int nVar [protected]

Definition at line 210 of file Opt.h.

4.30.4.31 vector<vector <Index> > nzhelements [protected]

nzero elements for the hessian matrix

Definition at line 225 of file Opt.h.

4.30.4.32 vector<vector <Index> > nzjelements [protected]

nzero elements for the jacobian matrix. nzelements[i][0] -> row (constrain), nzelements[i][1] -> column (variable)

Definition at line 224 of file Opt.h.

4.30.4.33 int options_g[4] [protected]

Definition at line 260 of file Opt.h.

4.30.4.34 int options_L[4] [protected]

Definition at line 261 of file Opt.h.

4.30.4.35 double overharvestingAllowance [protected]

Allows to harvest more than the resources available. Useful when resources got completelly exausted and the model refuses to solve.

Definition at line 220 of file Opt.h.

4.30.4.36 int previousYear [protected]

Definition at line 215 of file Opt.h.

4.30.4.37 vector<string> priPr [protected]

Definition at line 193 of file Opt.h.

4.30.4.38 vector< vector <int> > priPrCombs [protected]

A vector with all the possible combinations of primary products.

Definition at line 197 of file Opt.h.

4.30.4.39 unsigned int∗ rind_g [protected]

Definition at line 250 of file Opt.h.

4.30.4.40 unsigned int∗ rind_L [protected]

Definition at line 253 of file Opt.h.

4.30.4.41 unsigned int∗ rind_L_total [protected]

Definition at line 255 of file Opt.h.

4.30.4.42 int secondYear [protected]

Definition at line 217 of file Opt.h.

4.30.4.43 vector<string> secPr [protected]

Definition at line 194 of file Opt.h.

4.30.4.44 map<string, endvar> vars [protected]

List of variables in the model and their domain: pr product, sec prod, all products or all products over each subregion pair (exports)

Definition at line 202 of file Opt.h.

4.30.4.45 map<string, vector < vector < vector < vector <int> > > > > vpositions [protected]

cached position in the concatenated vector for each variables. Dimensions are l1reg, l2reg, prod, (l2To region).

Definition at line 203 of file Opt.h.

4.30.4.46 int worldCodeLev2 [protected]

Definition at line 218 of file Opt.h.

4.30.4.47 double∗ x_lam [protected]

Definition at line 246 of file Opt.h.

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

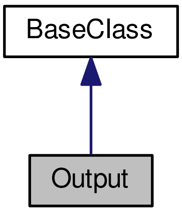
- /home/lobianco/git/ffsm_pp/src/Opt.h
- /home/lobianco/git/ffsm_pp/src/Opt.cpp

4.31 Output Class Reference

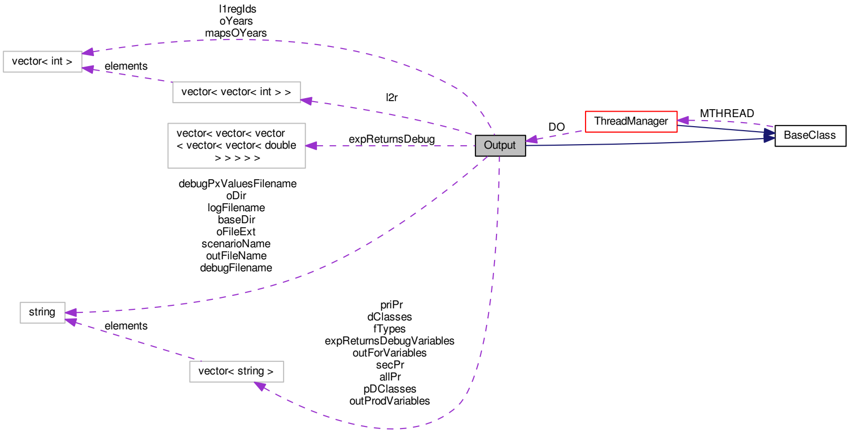
Output methods

`#include <Output.h>`

Inheritance diagram for Output:



Collaboration diagram for Output:



Public Member Functions

- Output (ThreadManager ∗MTHREAD_h)

  Constructor.
- ∼Output ()
- void initOutput ()
- void commonInit ()
- void initOutputMaps ()
- void initOutputForestData ()
- void initOutputProductData ()
- void initOptimisationLog ()
- void initDebugOutput ()

- void initDebugPixelValues ()
- void initCarbonBalance ()
- void print ()
- void printMaps ()
- void printForestData (bool finalFlush=false)
- void printProductData (bool finalFlush=false)
- void printCarbonBalance ()
- void printFinalOutput ()
- void printDebugOutput ()
- void printDebugPixelValues ()
- void printOptLog (bool optimal, int &nIterations, double &obj)
- char getOutputFieldDelimiter ()
- void cleanScenario (string fileName, string scenarioName, char d)

Public Attributes

- vector< vector< vector< vector< vector< double > > > > > expReturnsDebug

  l2_region, for type, d.c., pr prod, variable name
- vector< string > expReturnsDebugVariables

Private Attributes

- int oLevel
- char d
- int inYear
- int nYears
- string baseDir
- string oDir
- string scenarioName
- string oFileExt
- bool oHRedeable
- bool oSingleFile
- vector< int > oYears
- vector< int > mapsOYears
- int wRegId_l1
- int wRegId_l2
- string outFileName
- vector< string > outForVariables
- vector< string > outProdVariables
- int outStepRange
- bool forestDiamDetailedOutput
- vector< string > priPr
- vector< string > secPr
- vector< string > allPr
- vector< int > l1regIds
- vector< vector< int > > l2r
- vector< string > fTypes
- vector< string > dClasses
- vector< string > pDClasses

  includes an empty string for variables without diameter attribute
- int nPriPr
- int nSecPr
- int nAllPr
- int nL2r
- string logFilename
- string debugFilename
- string debugPxValuesFilename
- bool spMode

Additional Inherited Members

4.31.1 Detailed Description

Output methods

Class responsable to output the data, both as all kind of log as well as georeferenciated one.

Author

Antonello Lobianco

Definition at line 47 of file Output.h.

4.31.2 Constructor & Destructor Documentation

4.31.2.1 Output (ThreadManager ∗ *MTHREAD_h*)

Constructor.

Definition at line 37 of file Output.cpp.

```
00037                                          {
00038   MTHREAD=MTHREAD_h;
00039 }
```

4.31.2.2 ∼Output ( )

Definition at line 41 of file Output.cpp.

```
00041                   {
00042 }
```

4.31.3 Member Function Documentation

4.31.3.1 void cleanScenario (string *fileName,* string *scenarioName,* char *d*)

This routine clean the output scenario from previous outputs of the defined scenario. Other scenarios are untouched. The scenarioName must be in the first row.

Parameters

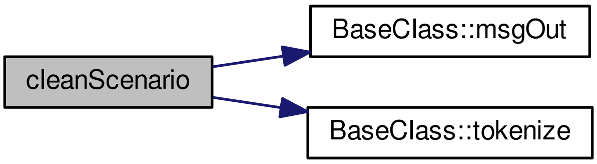
filename	Filename of the output file to clean
scenarioName	Name of the scenario we are replacing
d	Field delimiter. It must not be changed in the meantime (between the various scenarios)

Definition at line 951 of file Output.cpp.

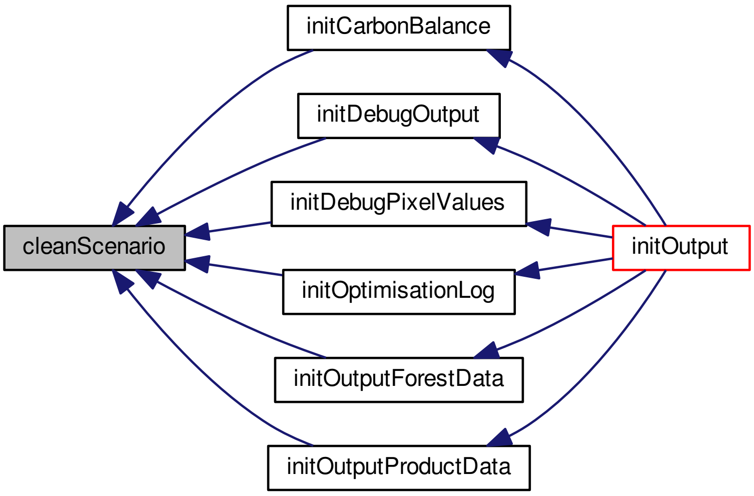
Referenced by initCarbonBalance(), initDebugOutput(), initDebugPixelValues(), initOptimisationLog(), initOutputForestData(), and initOutputProductData().

```
                                                                  {
  string dStr(&d,1);
  vector <string> rows;
  string tempRow;
  ifstream inFile (fileName.c_str(), ios::in);
  if (!inFile){
    msgOut(MSG_ERROR,"Error in opening the file "+fileName+" for reading.");
    return;
  }
  while( getline (inFile,tempRow) ){
    vector<string> tokens;
    tokenize(tempRow,tokens,dStr);
    if(tokens[0] != scenarioName)
      rows.push_back( tempRow );
  }
  inFile.close();
  ofstream out(fileName.c_str(), ios::out);
  for(uint i=0;i<rows.size();i++){
    out << rows[i];
    out << "\n";
  }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



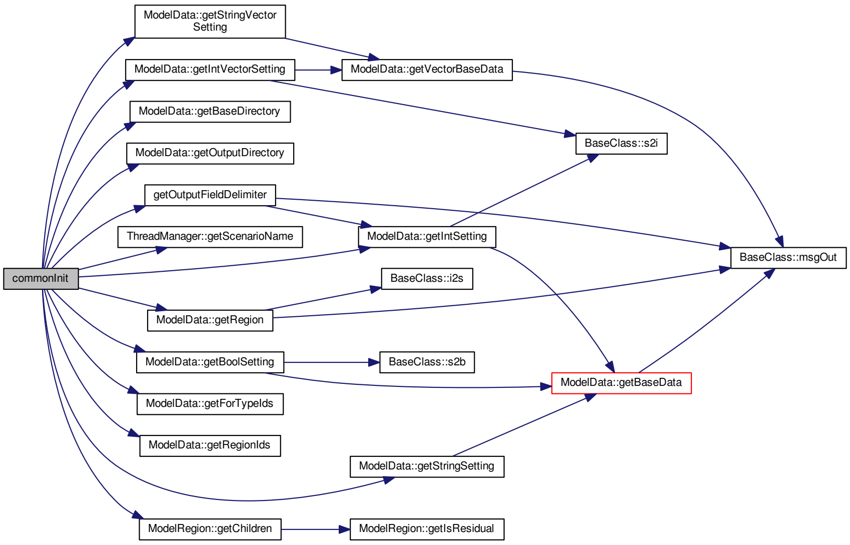
4.31.3.2 void commonInit ( )

Definition at line 61 of file Output.cpp.

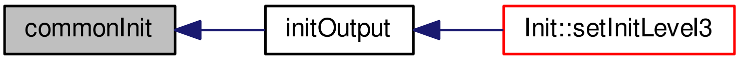
Referenced by initOutput().

```
00061                   {
00062   oLevel        = MTHREAD->MD->getIntSetting("outputLevel");
00063   d             = getOutputFieldDelimiter();
00064   inYear        = MTHREAD->MD->getIntSetting("initialYear");
00065   nYears        = MTHREAD->MD->getIntSetting("simulationYears");
00066   baseDir       = MTHREAD->MD->getBaseDirectory();
00067   oDir          = MTHREAD->MD->getOutputDirectory();
00068 //    bool initSeed = MTHREAD->MD->getBoolSetting("newRandomSeed");
00069 //    if (initSeed){
00070 //        uniform_int_distribution<> d(1, 1000000);
00071 //        int random = d(*MTHREAD->gen);
00072 //        scenarioName = MTHREAD->getScenarioName()+"_"+i2s(random);
00073 //    } else {
00074 //        scenarioName = MTHREAD->getScenarioName();
00075 //    }
00076   if (MTHREAD->MD->getStringSetting("overridenScenarioName") == "none"){
00077     scenarioName = MTHREAD->getScenarioName();
00078   } else {
00079     scenarioName = MTHREAD->MD->getStringSetting("
      overridenScenarioName");
00080   }
00081   oFileExt      = MTHREAD->MD->getStringSetting("outputFileExtension");
00082   oHRedeable    = MTHREAD->MD->getBoolSetting("outputHumanReadable");
00083   oSingleFile   = MTHREAD->MD->getBoolSetting("outputSingleFile");
00084   oYears        = MTHREAD->MD->getIntVectorSetting("outYears");
00085   mapsOYears    = MTHREAD->MD->getIntVectorSetting("mapsOutYears");
00086   wRegId_l1     = MTHREAD->MD->getIntSetting("worldCodeLev1");
00087   wRegId_l2     = MTHREAD->MD->getIntSetting("worldCodeLev2");
00088   outForVariables           = MTHREAD->MD->
      getStringVectorSetting("outForVariables");
00089   outProdVariables          = MTHREAD->MD->
      getStringVectorSetting("outProdVariables");
00090   dClasses                  = MTHREAD->MD->
      getStringVectorSetting("dClasses");
00091   pDClasses.insert(pDClasses.end(), dClasses.begin()+1,
      dClasses.end() ); // production diameter classes
00092   dClasses.push_back(""); // needed for reporting of variables without diameter attribute
00093   outStepRange              = MTHREAD->MD->getIntSetting("outStepRange");
00094   forestDiamDetailedOutput = MTHREAD->MD->
      getBoolSetting("forestDiamDetailedOutput");
00095   fTypes                    = MTHREAD->MD->getForTypeIds();
00096 
00097   priPr     = MTHREAD->MD->getStringVectorSetting("priProducts");
00098   secPr     = MTHREAD->MD->getStringVectorSetting("secProducts");
00099   allPr     = priPr;
00100   allPr.insert( allPr.end(), secPr.begin(), secPr.end() );
00101   nPriPr    = priPr.size();
00102   nSecPr    = secPr.size();
00103   nAllPr    = allPr.size();
00104   l1regIds = MTHREAD->MD->getRegionIds(1, true);
00105   nL2r      = MTHREAD->MD->getRegionIds(2, true).size();
00106   spMode    = MTHREAD->MD->getBoolSetting("usePixelData");
00107     //if(spMode) {
00108     //   pxIds =   getXyNPixels();
00109     //}
00110 
00111 
00112   for(uint i=0;i<l1regIds.size();i++){
00113     std::vector<int> l2ChildrenIds;
00114     ModelRegion* l1Region = MTHREAD->MD->getRegion(
      l1regIds[i]);
00115     std::vector<ModelRegion*> l2Childrens = l1Region->getChildren(true);
00116     for(uint j=0;j<l2Childrens.size();j++){
00117       l2ChildrenIds.push_back(l2Childrens[j]->getRegId());
00118     }
00119     if(l2ChildrenIds.size()){
00120       l2r.push_back(l2ChildrenIds);
00121     }
00122   }
00123 
00124 }
```

Here is the call graph for this function:


ModelData::getStringVector Setting
ModelData::getIntVectorSetting
ModelData::getVectorBaseData
ModelData::getBaseDirectory
BaseClass::s2i
ModelData::getOutputDirectory
getOutputFieldDelimiter
ThreadManager::getScenarioName
ModelData::getIntSetting
BaseClass::msgOut
commonInit
BaseClass::i2s
ModelData::getRegion
ModelData::getBoolSetting
BaseClass::s2b
ModelData::getBaseData
ModelData::getForTypeIds
ModelData::getRegionIds
ModelData::getStringSetting
ModelRegion::getChildren
ModelRegion::getIsResidual

Here is the caller graph for this function:



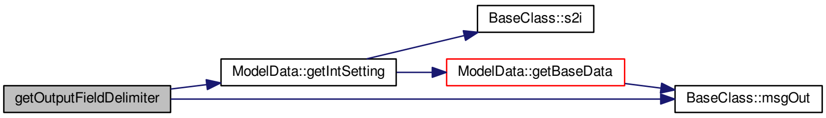
4.31.3.3 char getOutputFieldDelimiter ( )

Definition at line 780 of file Output.cpp.

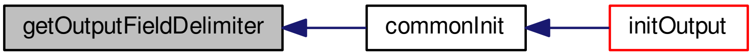
Referenced by commonInit().

```
00780                              {
00781   int delimiterID = MTHREAD->MD->getIntSetting("outputFieldDelimiter");
00782   switch (delimiterID) {
00783     case 1:
00784       return ',';
00785       break;
00786     case 2:
00787       return ';';
00788       break;
00789     case 3:
00790       return ':';
00791       break;
00792     case 4:
00793       return '\t';
00794       break;
00795     case 5:
00796       return ' ';
00797       break;
00798     default:
00799       msgOut(MSG_ERROR, "You have specified an unknow output file field delimiter. Using \";
      \".");
00800       return ',';
00801   }
00802 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



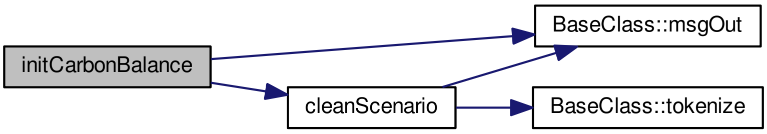
4.31.3.4 void initCarbonBalance ( )

Definition at line 348 of file Output.cpp.

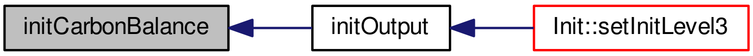
Referenced by initOutput().

```
00348                         {
00349
00350   if(oSingleFile){
00351     outFileName = baseDir+oDir+"results/carbonBalance"+
oFileExt;
00352     ifstream in(outFileName.c_str(), ios::in);
00353     if(in.is_open()) { // file exist, no need to initializate it, but we are gonna clean it of previous
 data of the same scenario if present...
00354       in.close();
00355       cleanScenario(outFileName, scenarioName,
d);
00356       return;
00357     } else {
00358       in.close();
00359     }
00360   } else {
00361     outFileName = baseDir+oDir+"results/carbonBalance_"+
scenarioName+oFileExt;
00362   }
00363
00364   ofstream out(outFileName.c_str(), ios::out);
00365   if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+
outFileName+" for reading.");}
00366     out << "scen" << d << "country" << d << "region" << d << "balItem" << d;
00367   //if(oHRedeable){
00368   //  for(int i=0;i<nYears;i++){
00369   //    out << i+inYear << d;
00370   //  }
00371   //} else {
00372     out << "year" << d << "value" << d;
00373   //}
00374   out << "\n";
00375   out.close();
00376 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



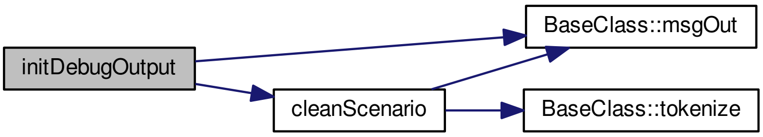
4.31.3.5 void initDebugOutput ( )

Definition at line 166 of file Output.cpp.

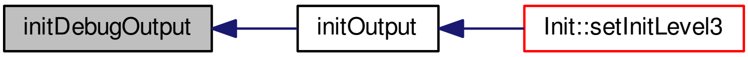
Referenced by initOutput().

```
00166                         {
00167     if(oLevel<OUTVL_ALL) return;
00168
00169     // init debugging the expected returns...
00170     if(spMode) return;
00171     expReturnsDebugVariables.push_back("hVol_byUPp");
00172     expReturnsDebugVariables.push_back("hV_byFT");
00173     expReturnsDebugVariables.push_back("finalHarvestFlag");
00174     expReturnsDebugVariables.push_back("pondCoeff");
00175     expReturnsDebugVariables.push_back("pW");
00176     expReturnsDebugVariables.push_back("cumTp");
00177     expReturnsDebugVariables.push_back("vHa");
00178     expReturnsDebugVariables.push_back("expectedReturns");
00179     expReturnsDebugVariables.push_back("weightedAvgCompModeFlag");
00180
00181     if (oSingleFile){
00182          debugFilename = baseDir+oDir+"debugs/debugOut.csv";
00183     } else {
00184          debugFilename = baseDir+oDir+"debugs/debugOut_"+
      scenarioName+".csv";
00185     }
00186
00187     ifstream in(debugFilename.c_str(), ios::in);
00188     if(in.is_open()) { // file exist, no need to initializate it, but we are gonna clean it of previous
       data of the same scenario if present...
00189         in.close();
00190         cleanScenario(debugFilename, scenarioName,
      d);
00191         return;
00192     } else { // file doesn't exist
00193         in.close();
00194         ofstream out(debugFilename.c_str(), ios::out);
00195         if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+
      debugFilename+" for writing.");}
00196         out << "scenario" << d << "year" << d << "region or pixel" << d << "forType" <<
      d << "freeDim" << d << "prod" << d << "parName" << d << "value" << d <<"\n";
00197         out.close();
00198     }
00199 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.31.3.6 void initDebugPixelValues ( )

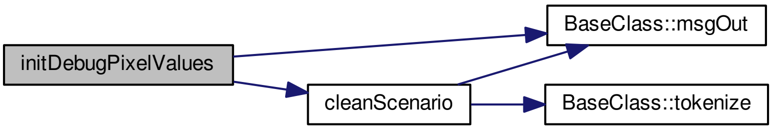
Definition at line 203 of file Output.cpp.

Referenced by initOutput().

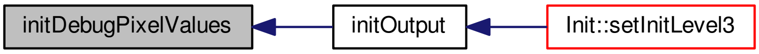
```
00203                            {
00204     if(oLevel<OUTVL_ALL) return;
00205 
00206     // init debugging the expected returns...
00207     if(!spMode) return;
00208 
00209     if (oSingleFile){
00210          debugPxValuesFilename = baseDir+oDir+"debugs/debugPxValues.csv";
00211     } else {
00212          debugPxValuesFilename = baseDir+oDir+"debugs/debugPxValues_"+
      scenarioName+".csv";
00213     }
00214 
00215     ifstream in(debugPxValuesFilename.c_str(), ios::in);
00216     if(in.is_open()) { // file exist, no need to initializate it, but we are gonna clean it of previous
       data of the same scenario if present...
00217         in.close();
00218         cleanScenario(debugPxValuesFilename,
      scenarioName, d);
00219         return;
00220     } else { // file doesn't exist
00221         in.close();
00222         ofstream out(debugPxValuesFilename.c_str(), ios::out);
00223         if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+
      debugPxValuesFilename+" for writing.");}
00224         out << "scenario" << d << "year" << d << "region" << d << "pxId" << d << "pxX" <<
      d << "pxY" << d ;
00225         for(uint f=0;f<fTypes.size();f++){
00226           string ft = fTypes[f];
00227           string header = "tp_multiplier_"+ft;
00228           out << header <<d;
00229         }
00230         for(uint f=0;f<fTypes.size();f++){
00231           string ft = fTypes[f];
00232           string header = "mortCoef_multiplier_"+ft;
00233           out << header <<d;
```

```
    }
    out << "var" << d ;

    for(uint f=0;f<fTypes.size();f++){
     string ft = fTypes[f];
     for (uint u=0;u<dClasses.size();u++){
      string dc=dClasses[u];
      string header = ft+"_"+dc;
      out << header <<d;
     }
    }
    out << "\n";


    out.close();
  }




  /*
  if(oSingleFile){
   outFileName = baseDir+oDir+"results/forestData"+oFileExt;
   ifstream in(outFileName.c_str(), ios::in);
   if(in.is_open()) { // file exist, no need to initializate it, but we are gonna clean it of previous data of the same scenario if present...
    in.close();
    cleanScenario(outFileName, scenarioName, d);
    return;
   } else {
    in.close();
   }
  } else {
   outFileName = baseDir+oDir+"results/forestData_"+scenarioName+oFileExt;
  }

  ofstream out(outFileName.c_str(), ios::out);
  if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+outFileName+" for reading.");}
   out << "scen" << d << "parName" << d << "country" << d << "region" << d << "forType" << d << "freeDim" << d;
  */









}
```

Here is the call graph for this function:



Here is the caller graph for this function:



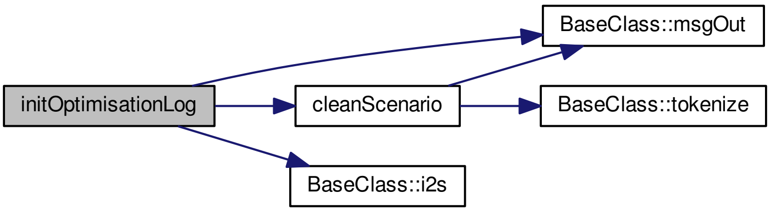
4.31.3.7 void initOptimisationLog ( )

Definition at line 127 of file Output.cpp.

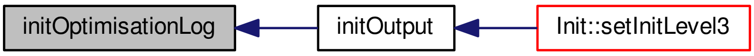
Referenced by initOutput().

```
00127                            {
00128   if(oLevel<OUTVL_AGGREGATED) return;
00129 
00130     if (oSingleFile){
00131          logFilename = baseDir+oDir+"optimisationLogs/optimisationLogs.txt";
00132 
00133     } else {
00134          logFilename = baseDir+oDir+"optimisationLogs/"+scenarioName+".txt";
00135     }
00136 
00137 
00138   ifstream in(logFilename.c_str(), ios::in);
00139   if(in.is_open()) { // file exist, no need to initializate it, but we are gonna clean it of previous data of the same scenario if present...
00140     in.close();
00141     cleanScenario(logFilename, scenarioName, d);
00142     ofstream out(logFilename.c_str(), ios::app);
00143     if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+logFilename+" for writing.");}
00144     time_t now;
00145     time(&now);
00146     struct tm *current = localtime(&now);
00147     string timemessage = i2s(current->tm_hour)+":"+i2s(current->tm_min)+":"+i2s(current->tm_sec);
00148     out << scenarioName << d << "0000" << d << timemessage << d << d << d <<"\n";
00149     out.close();
00150     return;
00151   } else { // file doesn't exist
00152     in.close();
00153     ofstream out(logFilename.c_str(), ios::out);
00154     if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+logFilename+" for writing.");}
00155     out << "scenario" << d << "year" << d << "time" << d << "opt flag" << d << "iterations" << d <<"\n";
00156     time_t now;
00157     time(&now);
00158     struct tm *current = localtime(&now);
00159     string timemessage = i2s(current->tm_hour)+":"+i2s(current->tm_min)+":"+i2s(current->tm_sec);
00160     out << scenarioName << d << "0000" << d << timemessage << d << d << d <<"\n";
00161     out.close();
00162   }
00163 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



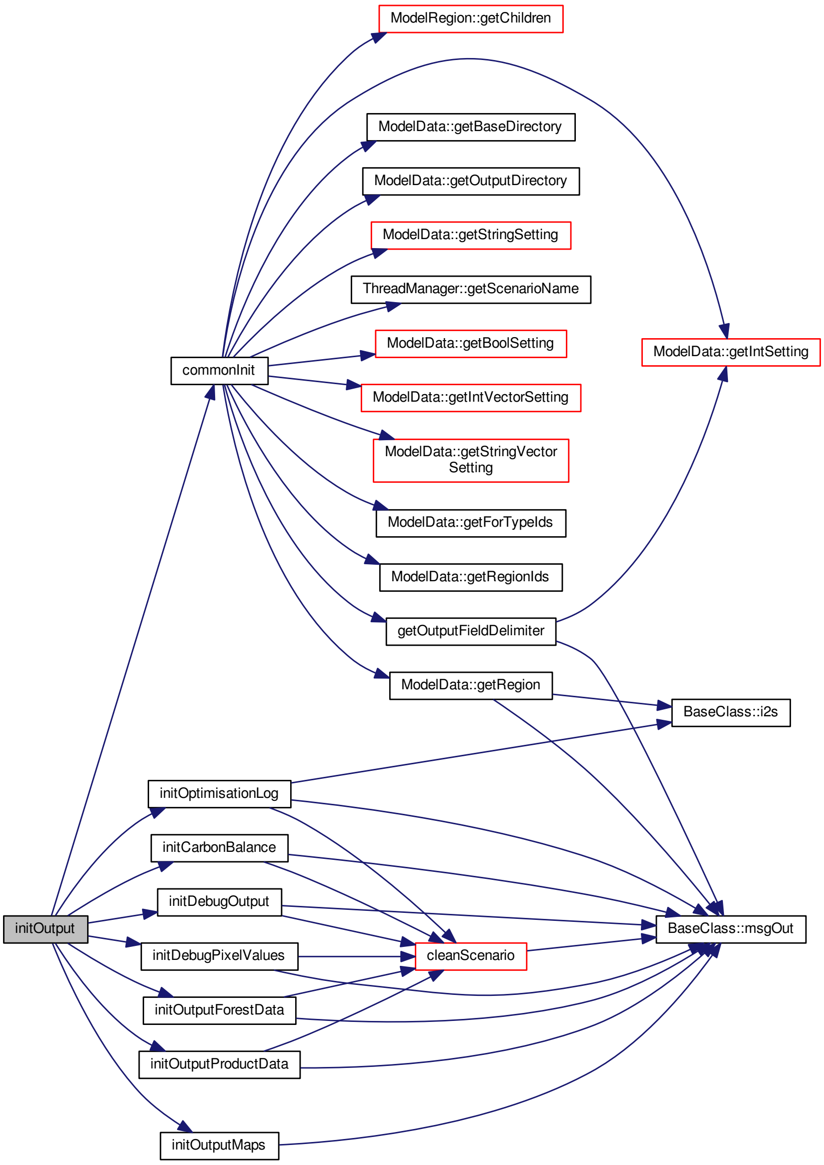
4.31.3.8 void initOutput ( )

Definition at line 48 of file Output.cpp.

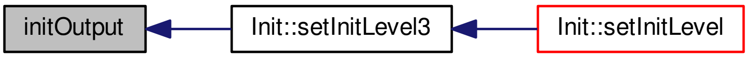
Referenced by Init::setInitLevel3().

```
00048                   {
00049   commonInit();
00050   initOutputMaps();
00051   initDebugOutput();
00052   initDebugPixelValues();
00053   initOutputForestData();
00054   initOutputProductData();
00055   initOptimisationLog();
00056   initCarbonBalance();
00057 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



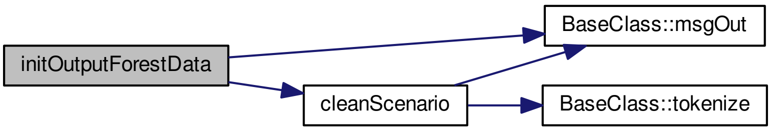
4.31.3.9 void initOutputForestData ( )

Definition at line 284 of file Output.cpp.

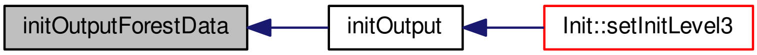
Referenced by initOutput().

```
                            {
  if(oLevel<OUTVL_DETAILED) return;

  if(oSingleFile){
    outFileName = baseDir+oDir+"results/forestData"+oFileExt;
    ifstream in(outFileName.c_str(), ios::in);
    if(in.is_open()) { // file exist, no need to initializate it, but we are gonna clean it of previous data of the same scenario if present...
      in.close();
      cleanScenario(outFileName, scenarioName, d);
      return;
    } else {
      in.close();
    }
  } else {
    outFileName = baseDir+oDir+"results/forestData_"+scenarioName+oFileExt;
  }

  ofstream out(outFileName.c_str(), ios::out);
  if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+outFileName+" for reading.");}
    out << "scen" << d << "parName" << d << "country" << d << "region" << d << "forType" << d << "freeDim" << d;
  if(oHRedeable){
    for(int i=0;i<nYears;i++){
      out << i+inYear << d;
    }
  } else {
    out << "year" << d << "value" << d;
  }
  out << "\n";
  out.close();
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.31.3.10 void initOutputMaps ( )

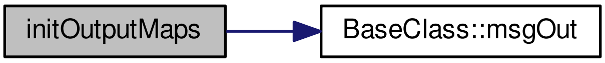
Resetting the list of printed layers and the scenario name..
Printing scenario name for post-processing scripts

Definition at line 384 of file Output.cpp.

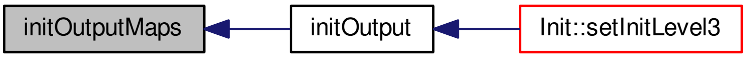
Referenced by initOutput().

```
00384                         {
00385   if(oLevel<OUTVL_MAPS) return;
00386   string mapBaseDirectory = baseDir+oDir+"maps/";
00387     string filenameToSaveScenarioName = mapBaseDirectory+"scenarioNames/"+
      scenarioName;
00388     string filenameListIntLayers = mapBaseDirectory+"integerListLayers/"+
      scenarioName;
00389     string filenameListFloatLayers = mapBaseDirectory+"floatListLayers/"+
      scenarioName;
00390
00391   // printing the scenario name in the "scenarioName file"...
00392   ofstream outSN(filenameToSaveScenarioName.c_str(), ios::out);
00393   if (!outSN){ msgOut(MSG_ERROR,"Error in opening the file "+filenameToSaveScenarioName+".")
      ;}
00394   outSN << scenarioName << "\n";
00395   outSN.close();
00396   // cleaning the "integerListLayers" and "floatListLayers" file...
00397   ofstream outi(filenameListIntLayers.c_str(), ios::out);
00398   outi.close();
00399   ofstream outf(filenameListFloatLayers.c_str(), ios::out);
00400   outf.close();
00401 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



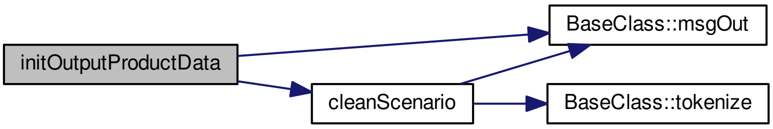
4.31.3.11 void initOutputProductData ( )

Definition at line 316 of file Output.cpp.

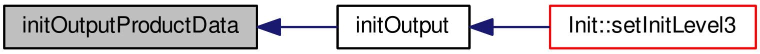
Referenced by initOutput().

```
00316                             {
00317   if(oLevel<OUTVL_DETAILED) return;
00318
00319   if(oSingleFile){
00320     outFileName = baseDir+oDir+"results/productData"+
      oFileExt;
00321     ifstream in(outFileName.c_str(), ios::in);
00322     if(in.is_open()) { // file exist, no need to initializate it, but we are gonna clean it of previous
       data of the same scenario if present...
00323       in.close();
00324       cleanScenario(outFileName, scenarioName,
      d);
00325       return;
00326     } else {
00327       in.close();
00328     }
00329   } else {
00330     outFileName = baseDir+oDir+"results/productData_"+
      scenarioName+oFileExt;
00331   }
00332
00333   ofstream out(outFileName.c_str(), ios::out);
00334   if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+
      outFileName+" for reading.");}
00335   out << "scen" << d << "parName" << d << "country" << d << "region" << d << "prod" <<
      d << "freeDim" << d;
00336   if(oHRedeable){
00337     for(int i=0;i<nYears;i++){
00338       out << i+inYear << d;
00339     }
00340   } else {
00341     out << "year" << d << "value" << d;
00342   }
00343   out << "\n";
00344   out.close();
00345 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



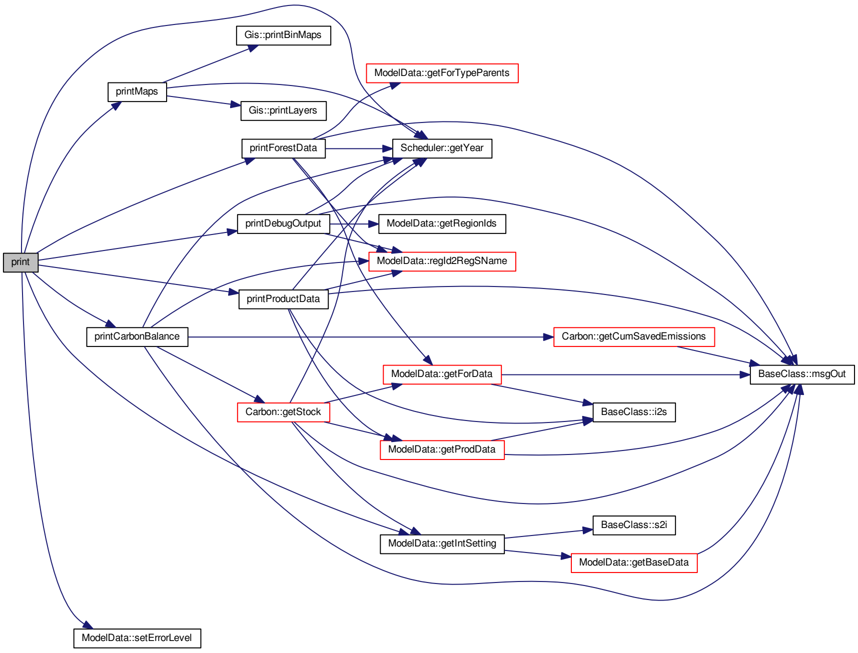
4.31.3.12 void print ( )

Definition at line 404 of file Output.cpp.

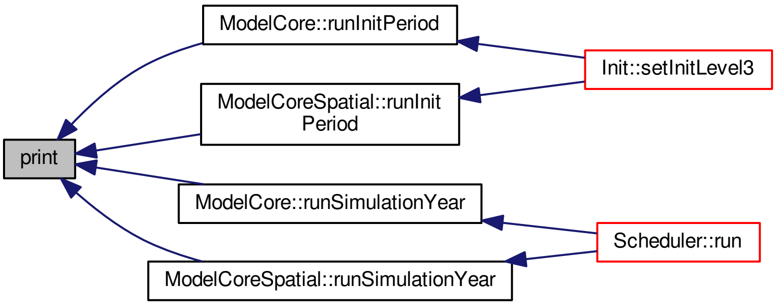
Referenced by ModelCore::runInitPeriod(), ModelCoreSpatial::runInitPeriod(), ModelCore::runSimulationYear(), and ModelCoreSpatial::runSimulationYear().

```
00404               {
00405   int cYear = MTHREAD->SCD->getYear();
00406   int initialSimulationYear = MTHREAD->MD->getIntSetting("initialOptYear");
00407
00408   if (outStepRange != -1 && (cYear-initialSimulationYear)%
      outStepRange != 0 && cYear != (initialSimulationYear+nYears)-1 ) {
00409     cout << cYear << " not printed" << endl;
00410     return;
00411   }
00412   bool printThisYear = false;
00413   for(uint i=0;i<oYears.size();i++){
00414     if (outStepRange == -1 && oYears[i] == cYear) printThisYear = true;
00415   }
00416   if(outStepRange == -1 && !printThisYear) return;
00417
00418
00419   cout << "printing " << cYear << endl;
00420   printMaps();
00421   MTHREAD->MD->setErrorLevel(MSG_NO_MSG);
00422   printForestData(false);
00423   printProductData(false);
00424   printCarbonBalance();
00425   printDebugOutput();
00426   MTHREAD->MD->setErrorLevel(MSG_ERROR);
00427 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


ModelCore::runInitPeriod
ModelCoreSpatial::runInit Period
print
ModelCore::runSimulationYear
ModelCoreSpatial::runSimulationYear
Init::setInitLevel3
Scheduler::run

4.31.3.13 void printCarbonBalance ( )

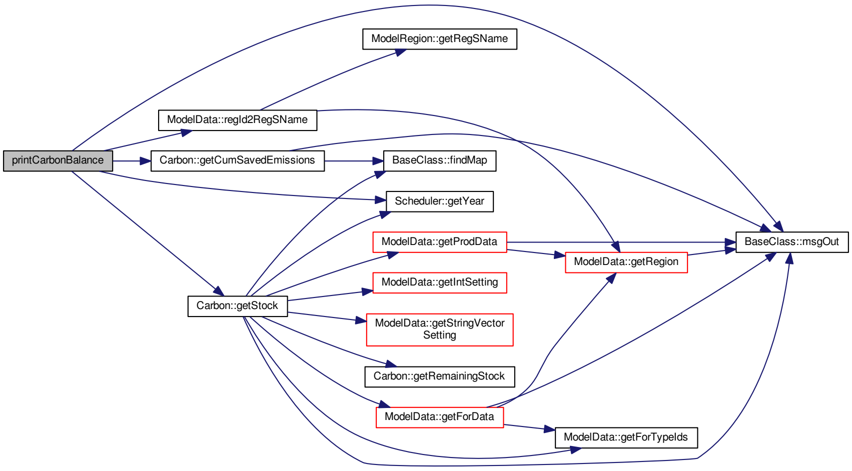
Definition at line 706 of file Output.cpp.

Referenced by print().

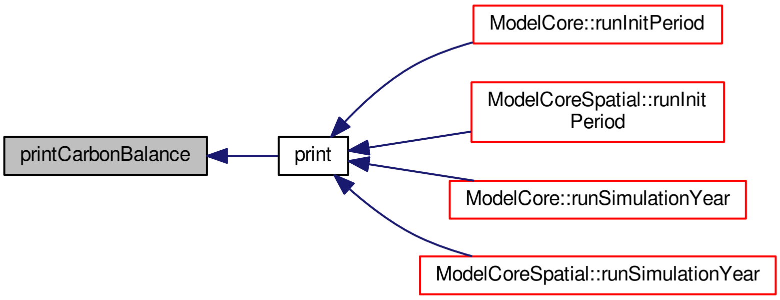
```
00706                            {
00707
00708   int currentYear = MTHREAD->SCD->getYear();
00709   if (currentYear == inYear) {return;} // don't print carbon balance on first year, carbon balance
 containers has not yet been initialised
00710
00711   msgOut(MSG_INFO, "Printing forest data..");
00712
00713   if(oSingleFile){
00714     outFileName = baseDir+oDir+"results/carbonBalance"+
oFileExt;
00715   } else {
00716     outFileName = baseDir+oDir+"results/carbonBalance_"+
scenarioName+oFileExt;
00717   }
00718   ofstream out (outFileName.c_str(), ios::app);
00719   if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+
outFileName+" for writing.");}
00720   double outvalue=0;
00721
00722   vector<int> balItems {STOCK_INV,STOCK_EXTRA,STOCK_PRODUCTS,
EM_ENSUB,EM_MATSUB,EM_FOROP};
00723
00724   for (uint r1=0;r1<l2r.size();r1++){
00725     for (uint r2=0;r2<l2r[r1].size();r2++){
00726       int regId = l2r[r1][r2];
00727       for (uint b=0;b<balItems.size();b++){
00728         out << scenarioName << d;
00729         out << MTHREAD->MD->regId2RegSName(l1regIds.at(r1)) << d;
00730         out << MTHREAD->MD->regId2RegSName(l2r[r1][r2]) <<
d;
00731         string balItemString;
00732         switch(balItems[b]){
00733           case STOCK_INV: {
00734             balItemString = "STOCK_INV";
00735             outvalue = MTHREAD->CBAL->getStock(regId, balItems[b]);
00736             break;
00737           }
00738           case STOCK_EXTRA: {
00739             balItemString = "STOCK_EXTRA";
00740             outvalue = MTHREAD->CBAL->getStock(regId, balItems[b]);
00741             break;
00742           }
00743           case STOCK_PRODUCTS: {
00744             balItemString = "STOCK_PRODUCTS";
00745             outvalue = MTHREAD->CBAL->getStock(regId, balItems[b]);
00746             break;
```

```
          }
          case EM_ENSUB: {
            balItemString = "EM_ENSUB";
            outvalue = MTHREAD->CBAL->getCumSavedEmissions(regId, balItems[b
]);
            break;
          }
          case EM_MATSUB: {
            balItemString = "EM_MATSUB";
            outvalue = MTHREAD->CBAL->getCumSavedEmissions(regId, balItems[b
]);
            break;
          }
          case EM_FOROP: {
            balItemString = "EM_FOROP";
            outvalue = MTHREAD->CBAL->getCumSavedEmissions(regId, balItems[b
]);
            break;
          }
        default:
            msgOut(MSG_CRITICAL_ERROR,"Unexpected balance item type in function
 printCarbonBalance");
        }
        out << balItemString << d;
        out << currentYear << d;
        out << outvalue << d;
        out << "\n";


      } // end bal items
    } // end r2
  } // end r1
  out.close();
}
```

Here is the call graph for this function:



Here is the caller graph for this function:


printCarbonBalance
print
ModelCore::runInitPeriod
ModelCoreSpatial::runInit Period
ModelCore::runSimulationYear
ModelCoreSpatial::runSimulationYear

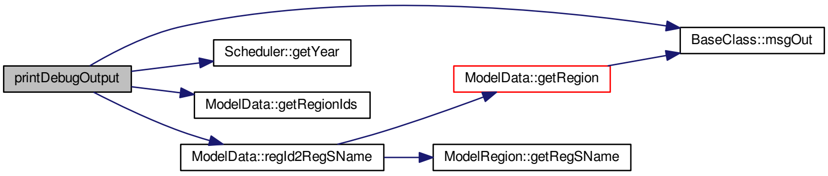
4.31.3.14 void printDebugOutput ( )

Definition at line 821 of file Output.cpp.

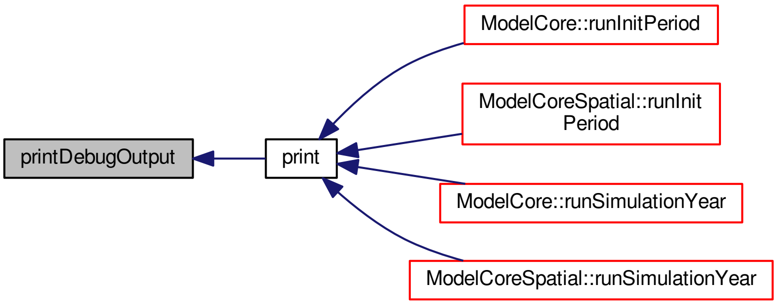
Referenced by print().

```
00821                           {
00822   if(oLevel<OUTVL_ALL) return;
00823
00824   // print debugging the expected returns...
00825
00826   if (!spMode && !expReturnsDebug.empty()){
00827     ofstream out (debugFilename.c_str(), ios::app);
00828     if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+
      debugFilename+" for writing.");}
00829     int currentYear = MTHREAD->SCD->getYear();
00830     vector <int> regIds2 = MTHREAD->MD->getRegionIds(2);
00831
00832         for (uint r2=0;r2<regIds2.size();r2++){
00833             for(uint ft=0;ft<fTypes.size();ft++){
00834                 for(uint dc=0;dc<(dClasses.size()-1);dc++){
00835                     for(uint pp=0;pp<priPr.size();pp++){
00836                         for(uint dv=0;dv<expReturnsDebugVariables.size();dv++){
00837                           // vector <vector < vector <vector <vector <double> > > > > expReturnsDebug;
00838                           double outputValue = expReturnsDebug.at(r2).at(ft).at(dc).at(pp).
      at(dv);
00839                           out << scenarioName << d;
00840                           out << currentYear << d;
00841                           out << MTHREAD->MD->regId2RegSName(regIds2[r2]) <<
      d;
00842                           out << fTypes[ft] << d;
00843                           out << dClasses[dc] << d;
00844                           out << priPr[pp] << d;
00845                           out << expReturnsDebugVariables[dv] <<
      d;
00846                           out << outputValue << d;
00847                           out << "\n";
00848                         }
00849                     }
00850                 }
00851             }
00852         }
00853
00854   } // end initial condition checks
00855 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


printDebugOutput
print
ModelCore::runInitPeriod
ModelCoreSpatial::runInit Period
ModelCore::runSimulationYear
ModelCoreSpatial::runSimulationYear

4.31.3.15 void printDebugPixelValues ( )

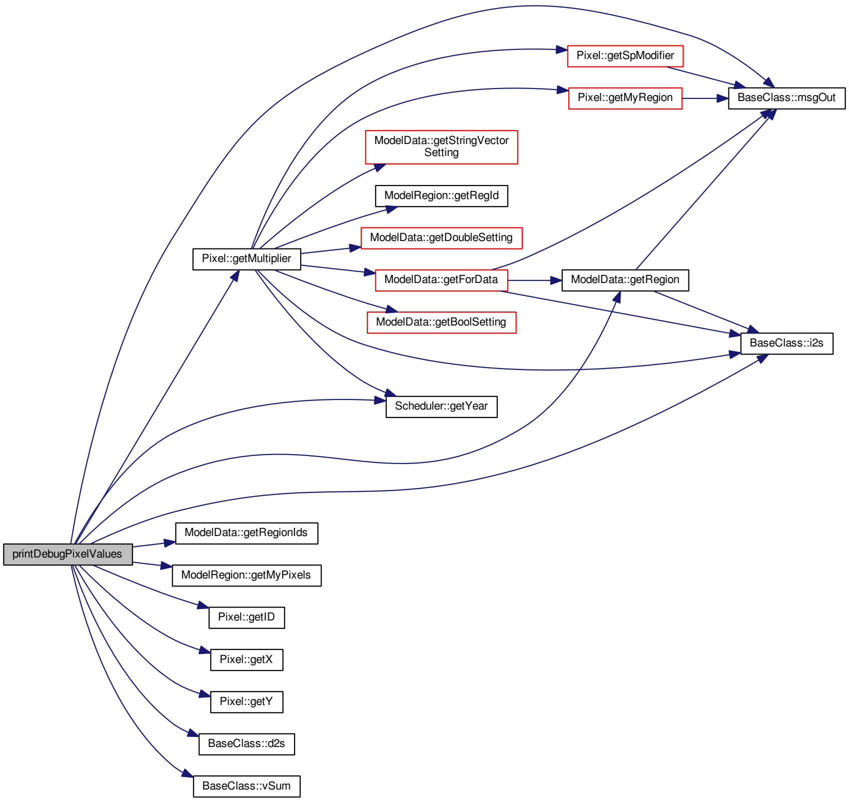
Definition at line 858 of file Output.cpp.

Referenced by ModelCoreSpatial::runInitPeriod(), and ModelCoreSpatial::runSimulationYear().

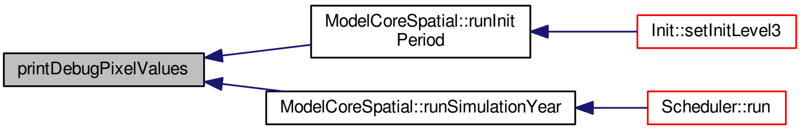
```
00858                              {
00859
00860   if(oLevel<OUTVL_ALL) return;
00861
00862   bool filter;
00863   filter = true; //use this to  filter output
00864   if(filter && spMode){
00865     ofstream out (debugPxValuesFilename.c_str(), ios::app);
00866     if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+
      debugPxValuesFilename+" for writing.");}
00867     int currentYear = MTHREAD->SCD->getYear();
00868     vector <int> regIds2 = MTHREAD->MD->getRegionIds(2);
00869     for (uint r=0;r<regIds2.size();r++){
00870       int rId = regIds2[r];
00871       //if(rId != 11061) continue;
00872       ModelRegion* REG = MTHREAD->MD->getRegion(rId);
00873       vector<Pixel*> regPx = REG->getMyPixels();
00874       for (uint p=0;p<regPx.size();p++){
00875         Pixel* px = regPx[p];
00876         int pxID = px->getID();
00877         int pxX = px->getX();
00878         int pxY = px->getY();
00879         string common = scenarioName + d + i2s(currentYear) + d +
      i2s(rId) + d+ i2s(pxID) +d +i2s(pxX)+d+i2s(pxY)+d;
00880
00881         for(uint f=0;f<fTypes.size();f++){
00882           double tp_m = px->getMultiplier("tp_multiplier",fTypes[f]);
```

```
          common += d2s(tp_m)+d;
        }
        for(uint f=0;f<fTypes.size();f++){
          double m_m = px->getMultiplier("mortCoef_multiplier", fTypes[f]);
          common += d2s(m_m)+d;
        }

        // First vars by only ft...
        // expectedReturns
        out << common << "expectedReturns" << d;
        for(uint f=0;f<fTypes.size();f++){
          for(uint u=0;u<dClasses.size()-1;u++){
            out << d;
          }
          out << px->expectedReturns[f] << d;
          //out << 0.0 << d;
        }
        out << "\n";
        //----
        out << common <<"vol" << d;
        for(uint f=0;f<fTypes.size();f++){
          for(uint u=0;u<dClasses.size()-1;u++){
            out << px->vol[f][u]<< d;
          }
          out << vSum(px->vol[f]) << d;
        }
        out << "\n";
        //----
        out << common <<"area" << d;
        for(uint f=0;f<fTypes.size();f++){
          for(uint u=0;u<dClasses.size()-1;u++){
            out << px->area[f][u]<< d;
          }
          out << vSum(px->area[f]) << d;
        }
        out << "\n";
        //----
        out << common <<"cumTp_exp" << d;
        for(uint f=0;f<fTypes.size();f++){
          for(uint u=0;u<dClasses.size()-1;u++){
            out << px->cumTp_exp[f][u]<< d;
          }
          out << vSum(px->cumTp_exp[f]) << d;
        }
        out << "\n";
        //----
        out << common <<"vHa_exp" << d;
        for(uint f=0;f<fTypes.size();f++){
          for(uint u=0;u<dClasses.size()-1;u++){
            out << px->vHa_exp[f][u]<< d;
          }
          out << vSum(px->vHa_exp[f]) << d;
        }
        out << "\n";
      } // end for each pixel
    } // end for each region
  } // end filter
} // end function printDebugPixelValues
```

Here is the call graph for this function:



Here is the caller graph for this function:


printDebugPixelValues
ModelCoreSpatial::runInit Period
Init::setInitLevel3
ModelCoreSpatial::runSimulationYear
Scheduler::run

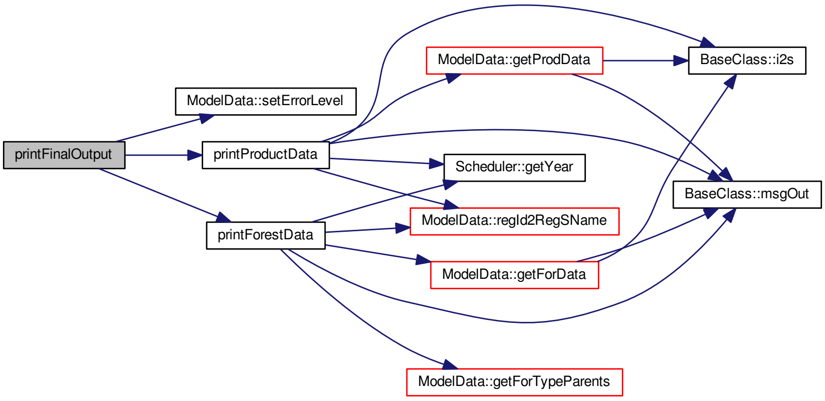
4.31.3.16 void printFinalOutput ( )

Definition at line 441 of file Output.cpp.

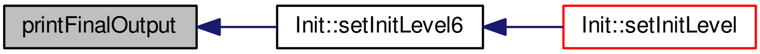
Referenced by Init::setInitLevel6().

```
00441                         {
00442   // we do this only if we choosed the outputHumanReadable settings, as we flush the data all in ones at
      the end.
00443   // oterwise we flush data every year
00444   if(oHRedeable){
00445     MTHREAD->MD->setErrorLevel(MSG_NO_MSG);
00446     printForestData(true);
00447     printProductData(true);
00448     MTHREAD->MD->setErrorLevel(MSG_ERROR);
00449   }
00450 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.31.3.17 void printForestData (bool *finalFlush* = `false`)

Definition at line 453 of file Output.cpp.

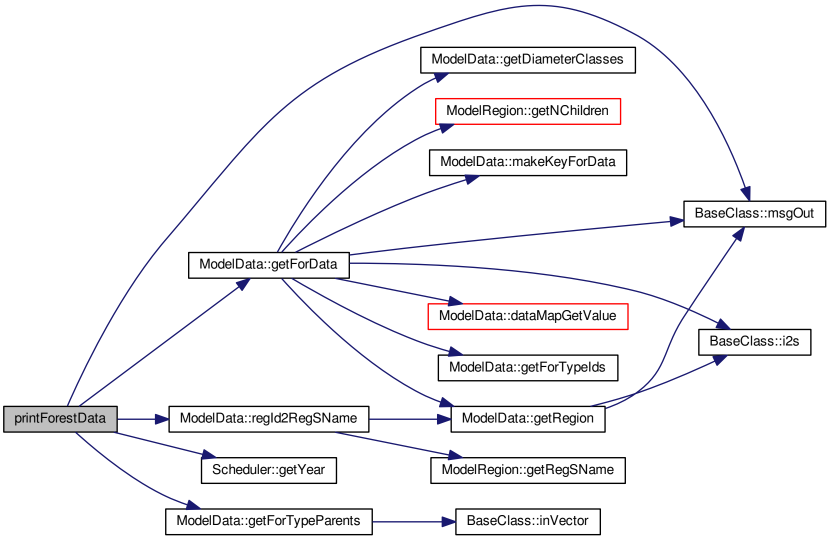
Referenced by ModelData::getCachedInitialYear(), print(), and printFinalOutput().

```
00453                                       {
00454 
00455   if(oLevel<OUTVL_DETAILED) return;
00456   if(oHRedeable && !finalFlush) return;
00457 
00458   msgOut(MSG_INFO, "Printing forest data..");
00459   int currentYear = MTHREAD->SCD->getYear();
00460   if(oSingleFile){
00461     outFileName = baseDir+oDir+"results/forestData"+
      oFileExt;
00462   } else {
00463     outFileName = baseDir+oDir+"results/forestData_"+
      scenarioName+oFileExt;
00464   }
00465   ofstream out (outFileName.c_str(), ios::app);
00466   if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+
      outFileName+" for writing.");}
00467   double outvalue;
00468   for(uint v=0;v<outForVariables.size();v++){
00469     vector<string>fTypes_temp = fTypes;
00470     if( outForVariables[v]=="expReturns" || outForVariables[v]=="
      sumExpReturns" ||  outForVariables[v]=="totalShareInvadedArea") {
00471       fTypes_temp.push_back(""); // adding an empty forest type to report for variables that doesn't have a
       forestType dimension
00472       vector<string> ftParents = MTHREAD->MD->getForTypeParents();
00473       fTypes_temp.insert(fTypes_temp.end(),ftParents.begin(),ftParents.end()); // also inserting forest
       type "parents" for expected returns
```

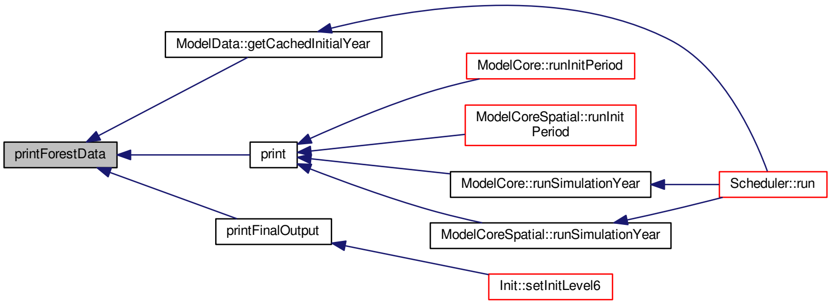
```
  }
  for (uint r1=0;r1<l2r.size();r1++){
   for (uint r2=0;r2<l2r[r1].size();r2++){
    for(uint ft=0;ft<fTypes_temp.size();ft++){
     if(forestDiamDetailedOutput){
      for(uint dc=0;dc<dClasses.size();dc++){ // an empty "" dc has been already added to the
 vector
       out << scenarioName << d;
       out << outForVariables[v] << d;
       out << MTHREAD->MD->regId2RegSName(l1regIds.at(r1)) << d;
       out << MTHREAD->MD->regId2RegSName(l2r[r1][r2]) <<
d;
       out << fTypes_temp[ft] << d;
       out << dClasses[dc] << d;
       if (oHRedeable){
        for(int y=0;y<nYears;y++){
         outvalue = MTHREAD->MD->getForData(
outForVariables[v],l2r[r1][r2],fTypes_temp[ft],dClasses[dc],y+
inYear);
         out << outvalue << d;
        }
        out << "\n";
       } else {
        outvalue = MTHREAD->MD->getForData(
outForVariables[v],l2r[r1][r2],fTypes_temp[ft],dClasses[dc]);
        out << currentYear << d;
        out << outvalue << d;
        out << "\n";
       }
      }
     } else {
      out << scenarioName << d;
      out << outForVariables[v] << d;
      out << MTHREAD->MD->regId2RegSName(l1regIds.at(r1)) << d;
      out << MTHREAD->MD->regId2RegSName(l2r[r1][r2]) <<
d;
      out << fTypes_temp[ft] << d;
      out << d;
      if (oHRedeable){
       for(int y=0;y<nYears;y++){
        outvalue = MTHREAD->MD->getForData(
outForVariables[v],l2r[r1][r2],fTypes_temp[ft],DIAM_ALL,y+
inYear);
        out << outvalue << d;
       }
       out << "\n";
      } else {
       outvalue = MTHREAD->MD->getForData(
outForVariables[v],l2r[r1][r2],fTypes_temp[ft],DIAM_ALL);
       out << currentYear << d;
       out << outvalue << d;
       out << "\n";
      }
     }
    }
   }
  }
 }
 /*
 DataMap::const_iterator i;
 string key;
 vector <double> values;
 string parName;
 int regId;
 string forType;
 string diamClass;
 for(i=MTHREAD->MD->forDataMap.begin();i!=MTHREAD->MD->forDataMap.end();i++){
  key = i->first;
  values = i->second;
  MTHREAD->MD->unpackKeyForData(key, parName, regId, forType, diamClass);
  ModelRegion* REG = MTHREAD->MD->getRegion(regId);
  // we don't want to output data from residual region unless it's the world region we are speaking of
  if(REG->getIsResidual() && !(regId==wRegId_l1 || regId==wRegId_l2)) continue;
  out << scenarioName << d;
  out << parName << d;
  if (REG->getRegLevel()==2){
   ModelRegion* pREG = MTHREAD->MD->getRegion(REG->getParRegId());
   out << pREG->getRegSName() << d;
   out << REG->getRegSName() << d;
  } else if (REG->getRegLevel()==1){
   out << REG->getRegSName() << d;
   out << d;
  } else {
   out << d << d;
  }
  out << forType << d;
  out << diamClass << d;
```

```
    if (oHRedeable){
      for(int y=0;y<nYears;y++){
        out << MTHREAD->MD->getTimedData(values,y+inYear) << d;
      }
      out << "\n";
    } else {
      out << currentYear << d;
      out << MTHREAD->MD->getTimedData(values,currentYear) << d;
      out << "\n";
    }
  }
  */
  out.close();
}
```

Here is the call graph for this function:



Here is the caller graph for this function:


ModelData::getCachedInitialYear
ModelCore::runInitPeriod
ModelCoreSpatial::runInit Period
printForestData
print
ModelCore::runSimulationYear
Scheduler::run
printFinalOutput
ModelCoreSpatial::runSimulationYear
Init::setInitLevel6

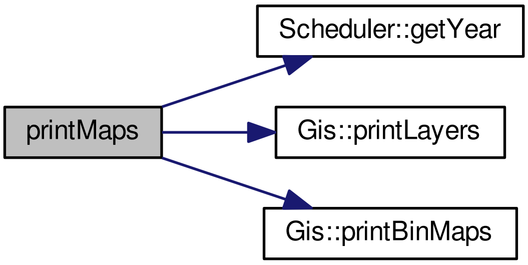
4.31.3.18 void printMaps ( )

Definition at line 430 of file Output.cpp.

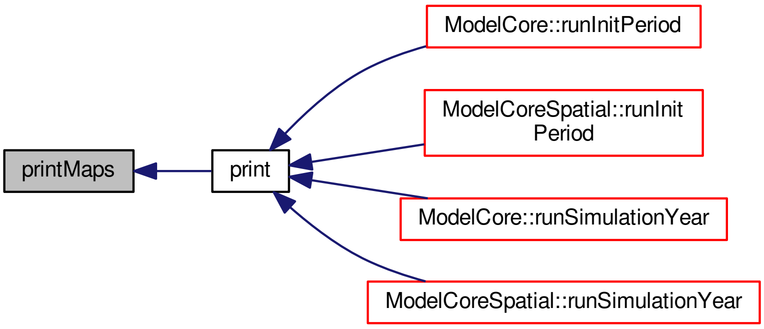
Referenced by print().

```
                  {
  if(oLevel<OUTVL_MAPS) return;
  int cYear = MTHREAD->SCD->getYear();
  if ( find(mapsOYears.begin(), mapsOYears.end(), cYear) !=
mapsOYears.end() ){
    MTHREAD->GIS->printLayers();
    if(oLevel<OUTVL_BINMAPS) return;
    MTHREAD->GIS->printBinMaps();
  }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:


ModelCore::runInitPeriod
ModelCoreSpatial::runInit Period
printMaps
print
ModelCore::runSimulationYear
ModelCoreSpatial::runSimulationYear

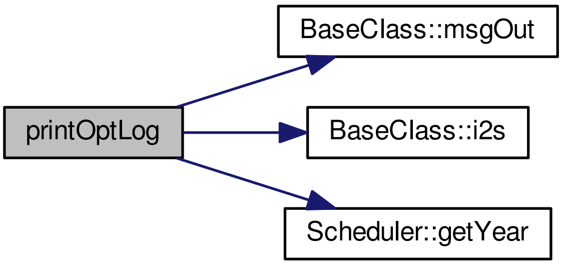
4.31.3.19 void printOptLog (bool *optimal,* int & *nIterations,* double & *obj*)

Definition at line 805 of file Output.cpp.

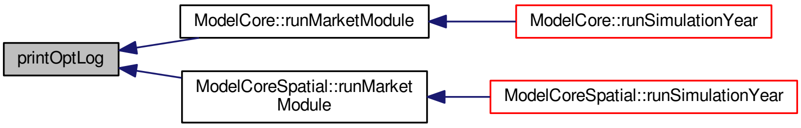
Referenced by ModelCore::runMarketModule(), and ModelCoreSpatial::runMarketModule().

```
00805                                                          {
00806   if(oLevel<OUTVL_AGGREGATED) return;
00807 
00808   ofstream out(logFilename.c_str(), ios::app);
00809   if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+
      logFilename+" for writing.");}
00810   time_t now;
00811   time(&now);
00812   struct tm *current = localtime(&now);
00813   string timemessage = i2s(current->tm_hour)+":"+i2s(current->tm_min)+":"+
      i2s(current->tm_sec);
00814   out << scenarioName << d << MTHREAD->SCD->getYear() <<
      d << timemessage << d << optimal;
00815   out << d << nIterations << d << obj << "\n";
00816   out.close();
00817 
00818 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


ModelCore::runMarketModule
ModelCore::runSimulationYear
printOptLog
ModelCoreSpatial::runMarket Module
ModelCoreSpatial::runSimulationYear

4.31.3.20 void printProductData (bool *finalFlush* = `false`)

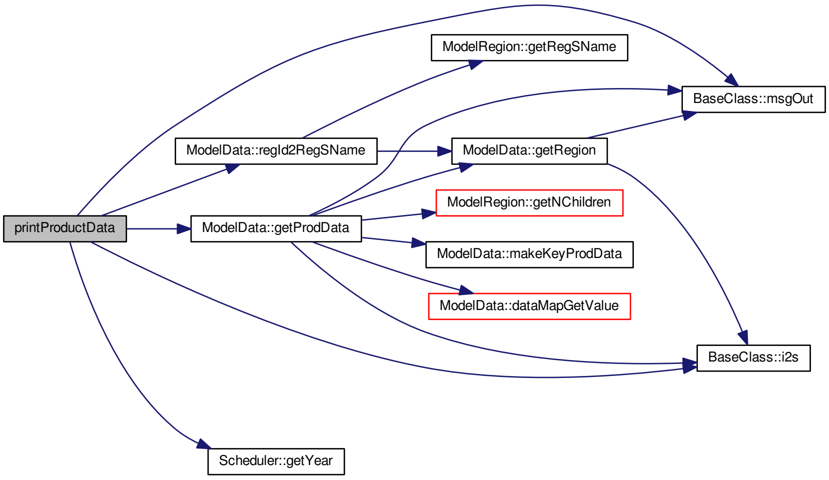
Definition at line 568 of file Output.cpp.

Referenced by ModelData::getCachedInitialYear(), print(), and printFinalOutput().

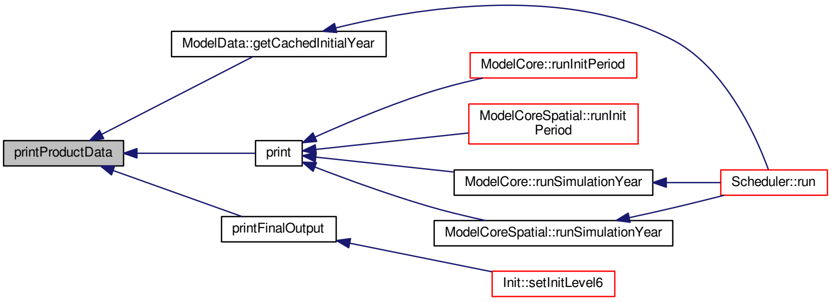
```
00568                                        {
00569 
00570   if(oLevel<OUTVL_DETAILED) return;
00571   if(oHRedeable && !finalFlush) return;
00572 
00573   msgOut(MSG_INFO, "Printing market data..");
00574   int currentYear = MTHREAD->SCD->getYear();
```

```

  if(oSingleFile){
    outFileName = baseDir+oDir+"results/productData"+
oFileExt;
  } else {
    outFileName = baseDir+oDir+"results/productData_"+
scenarioName+oFileExt;
  }
  ofstream out (outFileName.c_str(), ios::app);
  if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+
outFileName+" for writing.");}


  //11042  hardWSawnW  11083  0.00230651
  //11042  hardWSawnW  11082  0.0390874

  //if(MTHREAD->SCD->getYear() == 2007){
//  double test  = MTHREAD->MD->getProdData("rt",11042,"hardWSawnW",DATA_NOW);
//  double test2 = MTHREAD->MD->getProdData("rt",11042,"hardWSawnW",DATA_NOW,"11083");
//  double test3 = MTHREAD->MD->getProdData("rt",11042,"hardWSawnW",DATA_NOW,"11082");
//  cout << test << '\t' << test2 << '\t' << test3 << endl;
//  exit(0);
//  }

  double outvalue;
  for(uint v=0;v<outProdVariables.size();v++){
    for (uint r1=0;r1<l2r.size();r1++){
      for (uint r2=0;r2<l2r[r1].size();r2++){
        for(uint p=0;p<allPr.size();p++){

          if(outProdVariables[v]=="rt"){
            for(uint r2b=0;r2b<l2r[r1].size();r2b++){
              out << scenarioName << d;
              out << outProdVariables[v] << d;
              out << MTHREAD->MD->regId2RegSName(l1regIds.at(r1)) << d;
              out << MTHREAD->MD->regId2RegSName(l2r[r1][r2]) <<
d;
              out << allPr[p] << d;
              out << l2r[r1][r2b] << d;
              if (oHRedeable){
                for(int y=0;y<nYears;y++){
                  outvalue = MTHREAD->MD->getProdData(
outProdVariables[v],l2r[r1][r2],allPr[p],y+inYear,
i2s(l2r[r1][r2b]));
                  out << outvalue << d;
                }
                out << "\n";
              } else {
//                if(MTHREAD->SCD->getYear() == 2007 && l2r[r1][r2] == 11042 && allPr[p] == "hardWSawnW" &&
 (l2r[r1][r2b]== 11083 || l2r[r1][r2b]== 11082 )){
//                outvalue =
 MTHREAD->MD->getProdData(outProdVariables[v],l2r[r1][r2],allPr[p],currentYear,i2s(l2r[r1][r2b]));
//                cout << outvalue << endl;
//                }
                outvalue = MTHREAD->MD->getProdData(
outProdVariables[v],l2r[r1][r2],allPr[p],currentYear,i2s(
l2r[r1][r2b]));
                out << currentYear << d;
                out << outvalue << d;
                out << "\n";
              }
            }
          } else {
            out << scenarioName << d;
            out << outProdVariables[v] << d;
            out << MTHREAD->MD->regId2RegSName(l1regIds.at(r1)) << d;
            out << MTHREAD->MD->regId2RegSName(l2r[r1][r2]) <<
d;
            out << allPr[p] << d;
            out << d;
            if (oHRedeable){
              for(int y=0;y<nYears;y++){
                outvalue = MTHREAD->MD->getProdData(
outProdVariables[v],l2r[r1][r2],allPr[p],y+inYear);
                out << outvalue << d;
              }
              out << "\n";
            } else {
              outvalue = MTHREAD->MD->getProdData(
outProdVariables[v],l2r[r1][r2],allPr[p]);
              out << currentYear << d;
              out << outvalue << d;
              out << "\n";
            }

          }
        }
      }
    }
  }




/*
  DataMap::const_iterator i;
  string key;
  vector <double> values;
  string parName;
  int regId;
  string prod;
  string freeDim;
  for(i=MTHREAD->MD->prodDataMap.begin();i!=MTHREAD->MD->prodDataMap.end();i++){
    key = i->first;
    values = i->second;
    MTHREAD->MD->unpackKeyProdData(key, parName, regId, prod, freeDim);
    ModelRegion* REG = MTHREAD->MD->getRegion(regId);
    // we don't want to output data from residual region unless it's the world region we are speaking of
    if(REG->getIsResidual() && !(regId==wRegId_l1 || regId==wRegId_l2)) continue;
    out << scenarioName << d;
    out << parName << d;
    if (REG->getRegLevel()==2){
      ModelRegion* pREG = MTHREAD->MD->getRegion(REG->getParRegId());
      out << pREG->getRegSName() << d;
      out << REG->getRegSName() << d;
    } else if (REG->getRegLevel()==1){
      out << REG->getRegSName() << d;
      out << d;
    } else {
      out << d << d;
    }
    out << prod << d;
    out << freeDim << d;
    if (oHRedeable){
      for(int y=0;y<nYears;y++){
        out << MTHREAD->MD->getTimedData(values,y+inYear) << d;
      }
      out << "\n";
    } else {
      out << currentYear << d;
      out << MTHREAD->MD->getTimedData(values,currentYear) << d;
      out << "\n";
    }
  }

*/
  out.close();
}
```

Here is the call graph for this function:



Here is the caller graph for this function:


ModelData::getCachedInitialYear
ModelCore::runInitPeriod
ModelCoreSpatial::runInit Period
printProductData
print
ModelCore::runSimulationYear
Scheduler::run
printFinalOutput
ModelCoreSpatial::runSimulationYear
Init::setInitLevel6

4.31.4 Member Data Documentation

4.31.4.1 vector< string > allPr [private]

Definition at line 98 of file Output.h.

Referenced by commonInit(), and printProductData().

4.31.4.2 string baseDir [private]

Definition at line 81 of file Output.h.

Referenced by commonInit(), initCarbonBalance(), initDebugOutput(), initDebugPixelValues(), initOptimisationLog(), initOutputForestData(), initOutputMaps(), initOutputProductData(), printCarbonBalance(), printForestData(), and printProductData().

4.31.4.3 char d [private]

Definition at line 78 of file Output.h.

Referenced by commonInit(), initCarbonBalance(), initDebugOutput(), initDebugPixelValues(), initOptimisationLog(), initOutputForestData(), initOutputProductData(), printCarbonBalance(), printDebugOutput(), printDebugPixelValues(), printForestData(), printOptLog(), and printProductData().

4.31.4.4 vector< string > dClasses [private]

Definition at line 102 of file Output.h.

Referenced by commonInit(), initDebugPixelValues(), printDebugOutput(), printDebugPixelValues(), and printForestData().

4.31.4.5 string debugFilename [private]

Definition at line 109 of file Output.h.

Referenced by initDebugOutput(), and printDebugOutput().

4.31.4.6 string debugPxValuesFilename [private]

Definition at line 110 of file Output.h.

Referenced by initDebugPixelValues(), and printDebugPixelValues().

4.31.4.7 vector<vector < vector <vector <vector <double> > > > > expReturnsDebug

l2_region, for type, d.c., pr prod, variable name

Definition at line 73 of file Output.h.

Referenced by printDebugOutput(), and ModelCore::runManagementModule().

4.31.4.8 vector<string> expReturnsDebugVariables

Definition at line 74 of file Output.h.

Referenced by initDebugOutput(), and printDebugOutput().

4.31.4.9 bool forestDiamDetailedOutput [private]

Definition at line 95 of file Output.h.

Referenced by commonInit(), and printForestData().

4.31.4.10 vector<string> fTypes [private]

Definition at line 101 of file Output.h.

Referenced by commonInit(), initDebugPixelValues(), printDebugOutput(), printDebugPixelValues(), and printForestData().

4.31.4.11 int inYear [private]

Definition at line 79 of file Output.h.

Referenced by commonInit(), initOutputForestData(), initOutputProductData(), printCarbonBalance(), printForestData(), and printProductData().

4.31.4.12 vector<int> l1regIds [private]

Definition at line 99 of file Output.h.

Referenced by commonInit(), printCarbonBalance(), printForestData(), and printProductData().

4.31.4.13 vector< vector <int> > l2r [private]

Definition at line 100 of file Output.h.

Referenced by commonInit(), printCarbonBalance(), printForestData(), and printProductData().

4.31.4.14 string logFilename [private]

Definition at line 108 of file Output.h.

Referenced by initOptimisationLog(), and printOptLog().

4.31.4.15 vector<int> mapsOYears [private]

Definition at line 88 of file Output.h.

Referenced by commonInit(), and printMaps().

4.31.4.16 int nAllPr [private]

Definition at line 106 of file Output.h.

Referenced by commonInit().

4.31.4.17 int nL2r [private]

Definition at line 107 of file Output.h.

Referenced by commonInit().

4.31.4.18 int nPriPr [private]

Definition at line 104 of file Output.h.

Referenced by commonInit().

4.31.4.19 int nSecPr [private]

Definition at line 105 of file Output.h.

Referenced by commonInit().

4.31.4.20 int nYears [private]

Definition at line 80 of file Output.h.

Referenced by commonInit(), initOutputForestData(), initOutputProductData(), print(), printForestData(), and printProductData().

4.31.4.21 string oDir [private]

Definition at line 82 of file Output.h.

Referenced by commonInit(), initCarbonBalance(), initDebugOutput(), initDebugPixelValues(), initOptimisationLog(), initOutputForestData(), initOutputMaps(), initOutputProductData(), printCarbonBalance(), printForestData(), and printProductData().

4.31.4.22 string oFileExt [private]

Definition at line 84 of file Output.h.

Referenced by commonInit(), initCarbonBalance(), initOutputForestData(), initOutputProductData(), printCarbonBalance(), printForestData(), and printProductData().

4.31.4.23 bool oHRedeable [private]

Definition at line 85 of file Output.h.

Referenced by commonInit(), initOutputForestData(), initOutputProductData(), printFinalOutput(), printForestData(), and printProductData().

4.31.4.24 int oLevel [private]

Definition at line 77 of file Output.h.

Referenced by commonInit(), initDebugOutput(), initDebugPixelValues(), initOptimisationLog(), initOutputForestData(), initOutputMaps(), initOutputProductData(), printDebugOutput(), printDebugPixelValues(), printForestData(), printMaps(), printOptLog(), and printProductData().

4.31.4.25 bool oSingleFile [private]

Definition at line 86 of file Output.h.

Referenced by commonInit(), initCarbonBalance(), initDebugOutput(), initDebugPixelValues(), initOptimisationLog(), initOutputForestData(), initOutputProductData(), printCarbonBalance(), printForestData(), and printProductData().

4.31.4.26 string outFileName [private]

Definition at line 91 of file Output.h.

Referenced by initCarbonBalance(), initOutputForestData(), initOutputProductData(), printCarbonBalance(), printForestData(), and printProductData().

4.31.4.27 vector<string> outForVariables [private]

Definition at line 92 of file Output.h.

Referenced by commonInit(), and printForestData().

4.31.4.28 vector<string> outProdVariables [private]

Definition at line 93 of file Output.h.

Referenced by commonInit(), and printProductData().

4.31.4.29 int outStepRange [private]

Definition at line 94 of file Output.h.

Referenced by commonInit(), and print().

4.31.4.30 vector<int> oYears [private]

Definition at line 87 of file Output.h.

Referenced by commonInit(), and print().

4.31.4.31 vector<string> pDClasses [private]

includes an empty string for variables without diameter attribute

production diameter classes: exclude the fist diameter class below 15 cm

Definition at line 103 of file Output.h.

Referenced by commonInit().

4.31.4.32 vector<string> priPr [private]

Definition at line 96 of file Output.h.

Referenced by commonInit(), and printDebugOutput().

4.31.4.33 string scenarioName [private]

Definition at line 83 of file Output.h.

Referenced by commonInit(), initCarbonBalance(), initDebugOutput(), initDebugPixelValues(), initOptimisationLog(), initOutputForestData(), initOutputMaps(), initOutputProductData(), printCarbonBalance(), printDebugOutput(), printDebugPixelValues(), printForestData(), printOptLog(), and printProductData().

4.31.4.34 vector<string> secPr [private]

Definition at line 97 of file Output.h.

Referenced by commonInit().

4.31.4.35 bool spMode [private]

Definition at line 111 of file Output.h.

Referenced by commonInit(), initDebugOutput(), initDebugPixelValues(), printDebugOutput(), and printDebugPixelValues().

4.31.4.36 int wRegId_l1 [private]

Definition at line 89 of file Output.h.

Referenced by commonInit().

4.31.4.37 int wRegId_l2 [private]

Definition at line 90 of file Output.h.

Referenced by commonInit().

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

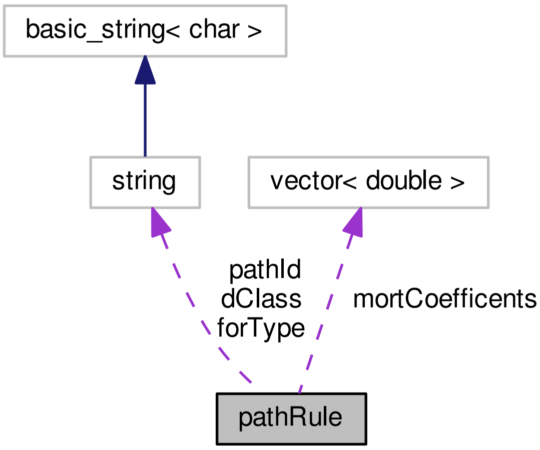
- /home/lobianco/git/ffsm_pp/src/Output.h
- /home/lobianco/git/ffsm_pp/src/Output.cpp

4.32 pathRule Struct Reference

Pathogen rule (how pathogen presense influence mortality) for a given forest type and diameter class (struct)

`#include <ModelData.h>`

Collaboration diagram for pathRule:



Public Attributes

- string forType
- string dClass
- string pathId

  Pathogen id (name)
- double pres_min

  Minimum level of presence of the pathogen to be counted as present (tolerance threshold)
- vector< double > mortCoefficents

  Mortality coefficients ordered by number of presence of the pathogen, e.g. first value is the mortality increase in the first year of pathogen appareance.

4.32.1 Detailed Description

Pathogen rule (how pathogen presense influence mortality) for a given forest type and diameter class (struct)

Struct containing the rule that affect the mortality of a given ft and dc by a given pathogen: depending on the number of year of presence of the pathogen over a given tollerance level the mortality increase more and more.

Definition at line 309 of file ModelData.h.

4.32.2 Member Data Documentation

4.32.2.1 string dClass

Definition at line 311 of file ModelData.h.

Referenced by ModelData::setDefaultPathogenRules(), and ModelData::setScenarioPathogenRules().

4.32.2.2 string forType

Definition at line 310 of file ModelData.h.

Referenced by ModelData::setDefaultPathogenRules(), and ModelData::setScenarioPathogenRules().

4.32.2.3 vector<double> mortCoefficents

Mortality coefficients ordered by number of presence of the pathogen, e.g. first value is the mortality increase in the first year of pathogen appareance.

Definition at line 314 of file ModelData.h.

Referenced by ModelData::setDefaultPathogenRules(), and ModelData::setScenarioPathogenRules().

4.32.2.4 string pathId

Pathogen id (name)

Definition at line 312 of file ModelData.h.

Referenced by ModelData::setDefaultPathogenRules(), and ModelData::setScenarioPathogenRules().

4.32.2.5 double pres_min

Minimum level of presence of the pathogen to be counted as present (tolerance threshold)

Definition at line 313 of file ModelData.h.

Referenced by ModelData::setDefaultPathogenRules(), and ModelData::setScenarioPathogenRules().

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

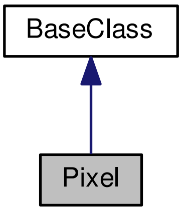
- /home/lobianco/git/ffsm_pp/src/ModelData.h

4.33 Pixel Class Reference

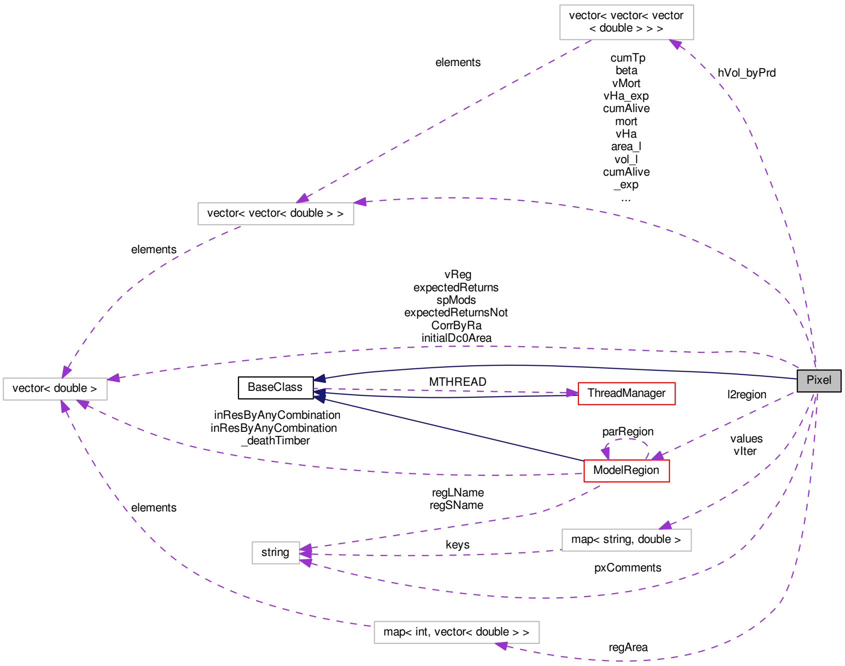
Pixel-level class.

`#include <Pixel.h>`

Inheritance diagram for Pixel:



Collaboration diagram for Pixel:



Public Member Functions

- Pixel (double ID_h, ThreadManager ∗MTHREAD_h)
- ∼Pixel ()
- double getDoubleValue (const string &layerName_h, const bool &returnZeroForNoValue=false) const

Return the value for a specific layer.

- double getDoubleValue (const string &parName, const string &forName, const string &dClass, const int &year, const bool &returnZeroForNoValue=false)
- double getMultiplier (const string &multiplierName, const string &forName, int year=DATA_NOW)
- double getPathMortality (const string &forType, const string &dC, int year=DATA_NOW)

  Return the INCREASED mortality due to pathogen presence for a given ft and dc in a certain year (default the running year)
- void correctInputMultiplier (const string &multiplierName, const string &forName, double coefficient=1)

  It apply a given coefficient to all the multipliers layers of a given ft.
- void newYear ()
- double getPastRegArea (const int &ft_idx, const int &year)
- void setPastRegArea (const double &value, const int &ft_idx, const int &year)
- ModelRegion ∗ getMyRegion (const int &rLevel=2)
- double getID () const
- int getX () const
- int getY () const
- vector< Pixel ∗ > getPixelsAtDistLevel (int distLevel_h) const

  Return a vector of pixels at the specified distance (in levels, not in physical units)
- string getPxComments () const
- double getCachedDouble () const
- void setValue (const string &layerName_h, const double &value_h)

  Insert a new layer and its value.
- void changeValue (const string &layerName_h, const double &value_h, const bool &setNoValueFor←Zero=false)

  Change the value of an existing layerMTHREAD->GIS->pack(parName, forName, dClass, year), value_h,.
- void setCoordinates (int x_h, int y_h)
- void setPxComments (std::string pxComments_h)
- void setCachedDouble (double cachedDouble_h)
- void clearCache ()
- void setSpModifier (const double &value, const int &ftindex)
- double getSpModifier (const string &ft)
- void swap (const int &swap_what)

  Assign to the delayed value the current values, e.g. vol_l = vol.
- void setMyRegion (ModelRegion ∗region_h)

Public Attributes

- vector< vector< double > > vol
- vector< vector< double > > area
- vector< double > initialDc0Area
- vector< vector< double > > hArea
- vector< vector< double > > hVol
- vector< vector< vector< double > > > hVol_byPrd
- map< int, vector< double > > regArea
- vector< double > vReg
- vector< vector< double > > vMort
- vector< double > expectedReturns
- vector< double > expectedReturnsNotCorrByRa

  by ft. Attenction, reported expReturns at "forest" level (compared with those at forest type level) do NOT include ra
- vector< vector< double > > vol_l

  store the volumes of the previous year
- vector< vector< double > > area_l

  store the areas of the previous year

- vector< vector< double > > beta
- vector< vector< double > > mort
- vector< vector< double > > tp
- vector< vector< double > > cumTp

  This is time of passage to REACH a diameter class (while the exogenous tp by diameter class is the time of passage to LEAVE to the next d class)
- vector< vector< double > > vHa

  Volume at hectar by each diameter class [m^3/ha].
- vector< vector< double > > cumAlive

  Cumulative prob of remaining alive at beginnin of a given diam class.
- vector< vector< double > > cumTp_exp

  *This is the **expected** version of cumTp, used for calculating profits.*
- vector< vector< double > > vHa_exp

  *This is the **expected** version of vHa, used for calculating profits.*
- vector< vector< double > > cumAlive_exp

  *This is the **expected** version of cumAlive, used for calculating profits.*
- double portfolioVarRa

  Sampling derived risk aversion on portfolio variance for of this agent.
- double expType

  Sampling derived expectation types of this agent (forest bilogical parameters: growth, mortality)
- double expTypePrices

  Sampling derived expectation types of this agent (prices)
- bool usePortfolio

  Sampling derived usage of portfolio management (false/true)
- double avalCoef

  Availability (of wood resources) coefficient. A [0,1] coefficient that reduce avaiability of wood resources to exploitation due to local reasons (protected area, altimetry..)

Private Attributes

- map< string, double > values

  Map of values for each layer.
- map< string, double >::const_iterator vIter
- double ID
- int pxX
- int pxY
- string pxComments
- double cachedDouble

  Cachable double used in some optimized algorithms.
- vector< double > spMods

  The sampled spatial modifiers (by forest type)
- ModelRegion ∗ l2region

  Pointer to level 2 region where this pixel is.

Additional Inherited Members

4.33.1 Detailed Description

Pixel-level class.

This class manage the info at the pixel level. A vector of pixel objects is owned by the class Gis.

Author

Antonello Lobianco

Definition at line 47 of file Pixel.h.

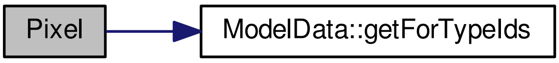
4.33.2 Constructor & Destructor Documentation

4.33.2.1 Pixel (double *ID_h,* ThreadManager ∗ *MTHREAD_h*)

Definition at line 27 of file Pixel.cpp.

```
00027                                                   : ID(ID_h)
00028 {
00029   MTHREAD=MTHREAD_h;
00030   int nft = MTHREAD->MD->getForTypeIds().size();
00031   vector<double> temp(nft,1);
00032   //vector<double> temp2(nft,0);
00033   spMods = temp;
00034   avalCoef = 1;
00035   //vMort  = temp2;
00036   //std::fill(v.begin(), v.end(), 0);
00037 }
```

Here is the call graph for this function:



4.33.2.2 ∼Pixel ( )

Definition at line 39 of file Pixel.cpp.

```
00040 {
00041 }
```

4.33.3 Member Function Documentation

4.33.3.1 void changeValue (const string & *layerName_h,* const double & *value_h,* const bool & *setNoValueForZero* = `false`)

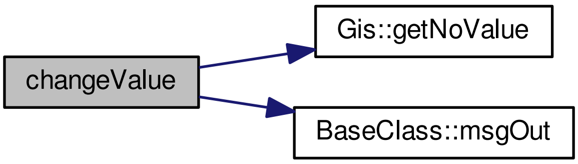
Change the value of an existing layerMTHREAD->GIS->pack(parName, forName, dClass, year), value_h,.

Definition at line 135 of file Pixel.cpp.

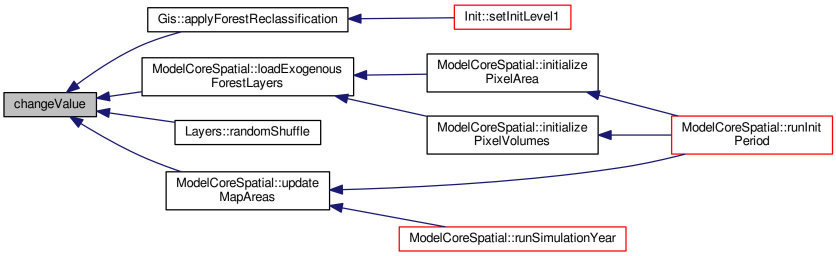
Referenced by Gis::applyForestReclassification(), ModelCoreSpatial::loadExogenousForestLayers(), Layers::randomShuffle(), and ModelCoreSpatial::updateMapAreas().

```
00135                                                                                                   {
00136   map<string, double>::iterator p;
00137   p=values.find(layerName_h);
00138   if(p != values.end()){
00139     if(setNoValueForZero && value_h == 0){
00140       p->second = MTHREAD->GIS->getNoValue();
00141     } else {
00142       p->second = value_h;
00143     }
00144   } else {
00145     msgOut(MSG_ERROR, "Coud not change pixel value for layer "+layerName_h+". Layer don't
       found.");
00146   }
00147   return;
00148 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


changeValue
Gis::applyForestReclassification
Init::setInitLevel1
ModelCoreSpatial::loadExogenous ForestLayers
ModelCoreSpatial::initialize PixelArea
ModelCoreSpatial::initialize PixelVolumes
ModelCoreSpatial::runInit Period
Layers::randomShuffle
ModelCoreSpatial::update MapAreas
ModelCoreSpatial::runSimulationYear

4.33.3.2 void clearCache ( ) `[inline]`

Definition at line 85 of file Pixel.h.

```
00085 {cachedDouble=0;};
```

4.33.3.3 void correctInputMultiplier (const string & *multiplierName,* const string & *forName,* double *coefficient* = `1`)

It apply a given coefficient to all the multipliers layers of a given ft.

Definition at line 275 of file Pixel.cpp.

```
00275                                                                                                     {
00276   string search_for = multiplierName+"#"+forName+"#";
00277   for (std::map<string,double>::iterator it=values.lower_bound(search_for); it!=
      values.end(); ++it){
00278     if (it->first.compare(0, search_for.size(), search_for) == 0){
00279             //cout << ID << ";" << forName << ";" << coefficient << endl;
00280       it->second = it->second * coefficient;
00281     }
00282   }
00283 }
```

4.33.3.4 double getCachedDouble ( ) const `[inline]`

Definition at line 73 of file Pixel.h.

```
00073 {return cachedDouble;};
```

4.33.3.5 double getDoubleValue (const string & *layerName_h,* const bool & *returnZeroForNoValue* = `false`) const

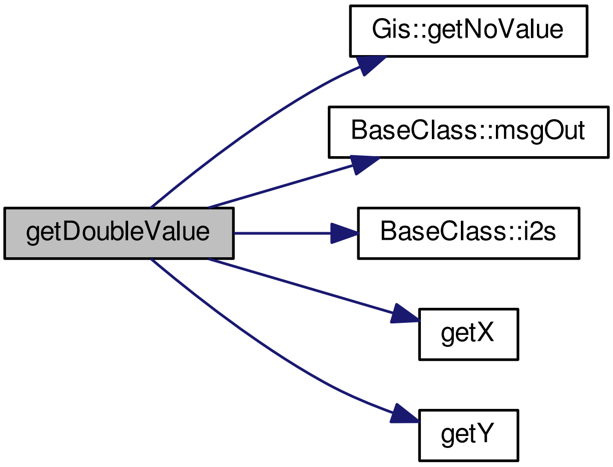
Return the value for a specific layer.

Definition at line 158 of file Pixel.cpp.

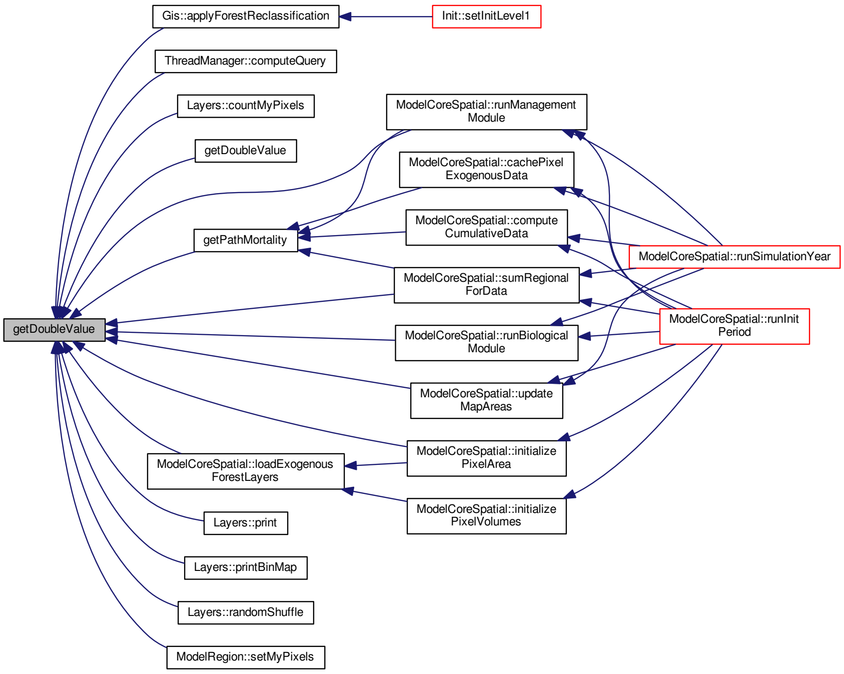
Referenced by Gis::applyForestReclassification(), ThreadManager::computeQuery(), Layers::countMyPixels(), getDoubleValue(), getPathMortality(), ModelCoreSpatial::initializePixelArea(), ModelCoreSpatial::loadExogenousForestLayers(), Layers::print(), Layers::printBinMap(), Layers::randomShuffle(), ModelCoreSpatial::runBiologicalModule(), ModelCoreSpatial::runManagementModule(), ModelRegion::setMyPixels(), ModelCoreSpatial::sumRegionalForData(), and ModelCoreSpatial::updateMapAreas().

```
00158                                                                                         {
00159   vIter=values.find(layerName_h);
00160   if(vIter != values.end()) {
00161     if(returnZeroForNoValue && vIter->second==MTHREAD->GIS->
      getNoValue()){
00162       return 0.0;
00163     } else {
00164       return vIter->second;
00165     }
00166   } else {
00167     msgOut(MSG_WARNING, "No layer \""+layerName_h+"\" found on pixel ("+
      i2s(getX())+","+i2s(getY())+"). Sure you didn't mispelled it?");
00168     if(returnZeroForNoValue){
00169       return 0.0;
00170     } else {
00171       return MTHREAD->GIS->getNoValue();
00172     }
00173   }
00174 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

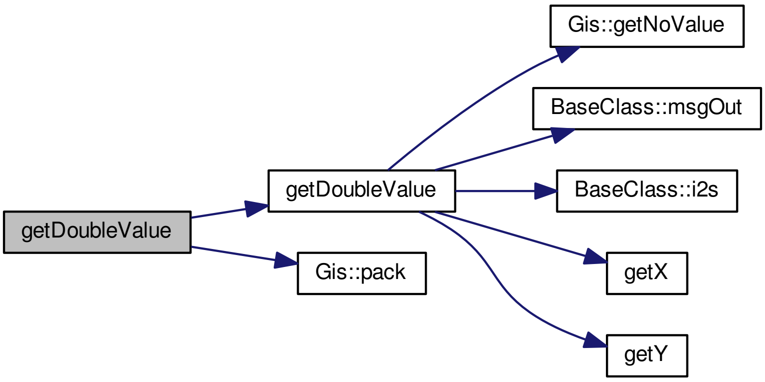

Gis::applyForestReclassification
Init::setInitLevel1
ThreadManager::computeQuery
Layers::countMyPixels
ModelCoreSpatial::runManagement Module
getDoubleValue
ModelCoreSpatial::cachePixel ExogenousData
ModelCoreSpatial::compute CumulativeData
getPathMortality
ModelCoreSpatial::runSimulationYear
ModelCoreSpatial::sumRegional ForData
ModelCoreSpatial::runInit Period
getDoubleValue
ModelCoreSpatial::runBiological Module
ModelCoreSpatial::update MapAreas
ModelCoreSpatial::initialize PixelArea
ModelCoreSpatial::loadExogenous ForestLayers
ModelCoreSpatial::initialize PixelVolumes
Layers::print
Layers::printBinMap
Layers::randomShuffle
ModelRegion::setMyPixels

4.33.3.6 double getDoubleValue (const string & *parName,* const string & *forName,* const string & *dClass,* const int & *year,* const bool & *returnZeroForNoValue* = `false`)

Definition at line 286 of file Pixel.cpp.

```
00286                                                    {
00287   return getDoubleValue(MTHREAD->GIS->pack(parName, forName, dClass, year),
      returnZeroForNoValue);
00288 }
```

Here is the call graph for this function:



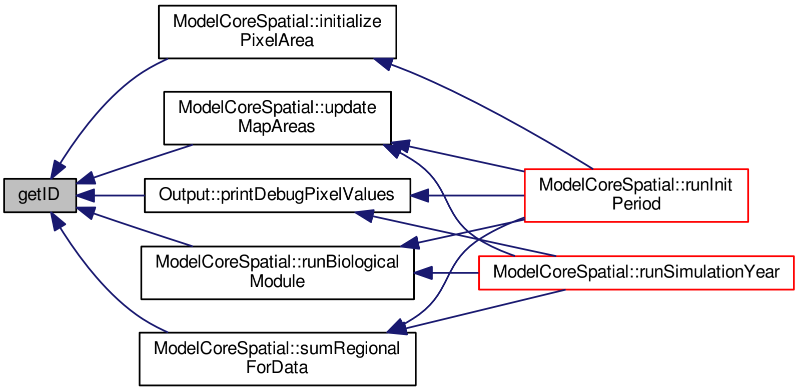
4.33.3.7 double getID ( ) const [inline]

Definition at line 66 of file Pixel.h.

Referenced by ModelCoreSpatial::initializePixelArea(), Output::printDebugPixelValues(), ModelCoreSpatial::runBiologicalModule(), ModelCoreSpatial::sumRegionalForData(), and ModelCoreSpatial::updateMapAreas().

```
00066 {return ID;}  ;
```

Here is the caller graph for this function:


ModelCoreSpatial::initialize PixelArea
ModelCoreSpatial::update MapAreas
getID
Output::printDebugPixelValues
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runBiological Module
ModelCoreSpatial::runSimulationYear
ModelCoreSpatial::sumRegional ForData

4.33.3.8 double getMultiplier (const string & *multiplierName,* const string & *forName,* int *year* = DATA_NOW)

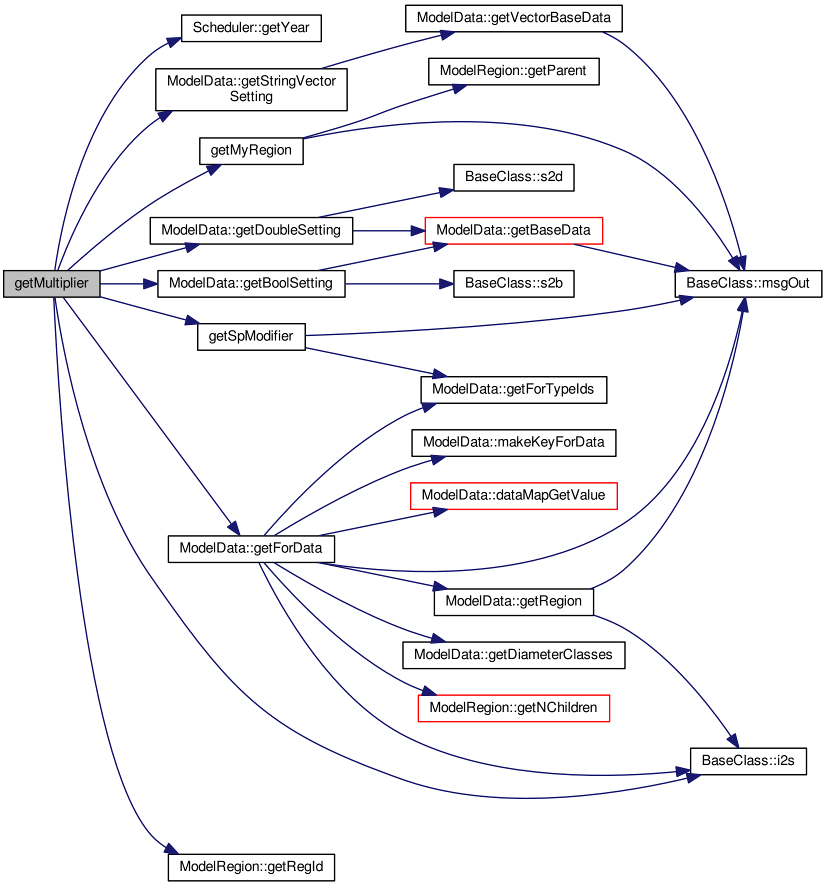
getMultiplier() returns the value of the multiplier as memorized in the spatialDataSubfolder layers or in the forData table. It will looks for exact match or for lower years if available. If no layers are available or the usePixelData option is not used, it will return 1. If the tp_multiplier is asked for, it will adjusts for spatial variance coefficient. If the mortCoef_multiplier is used and we are in the table settings it will adjust it by mortCoef_link.

Definition at line 184 of file Pixel.cpp.

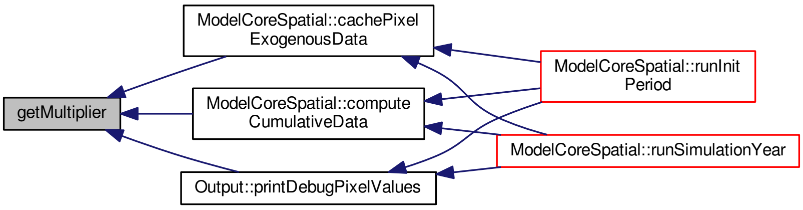
Referenced by ModelCoreSpatial::cachePixelExogenousData(), ModelCoreSpatial::computeCumulativeData(), and Output::printDebugPixelValues().

```
00184                                                                                   {
00185 
00186 
00187   if(year==DATA_NOW){year = MTHREAD->SCD->getYear();}
00188 
00189 
00190     double multiplierSpVar = (multiplierName == "tp_multiplier")?getSpModifier(forName):1.0;
00191 
00192     vector <string> modifiersFromTable = MTHREAD->MD->
getStringVectorSetting("modifiersFromTable");
00193 
00194     if(std::find(modifiersFromTable.begin(), modifiersFromTable.end(), multiplierName) !=
modifiersFromTable.end()) {
00195       // load multiplier from forData table..
00196       int regId = getMyRegion()->getRegId();
00197       double multiplier = MTHREAD->MD->getForData(multiplierName, regId, forName, "",
year);
00198       if (multiplierName == "mortCoef_multiplier"){
00199         return pow(multiplier,MTHREAD->MD->getDoubleSetting("mortMultiplier_link")
)*multiplierSpVar; //Added to account that our multipliers are based on probability of presence and not on
planted/managed forests, where mortality is somhow reduced
      }
      return multiplier*multiplierSpVar;

    } else {
      // load multiplier from layer file..

      // return 1 if not using pixel mode
      if(!MTHREAD->MD->getBoolSetting("usePixelData")) return 1.0;
      string search_for = multiplierName+"#"+forName+"##"+i2s(year);
      map <string,double>::const_iterator i = values.upper_bound(search_for); //return the position always upper to the found one, even if it's an equal match.
      if(i!= values.begin())  i--; // this rewind the position to the one just before or equal
      const string& key = i->first;
      string search_base = search_for.substr(0,search_for.size()-4);
      if (key.compare(0, search_base.size(), search_base) == 0){
          //cout << "MATCH: " << search_for <<", "<< i->first << ", " << i->second << endl;
          //if(i->second != 1){
          //  cout << "NOT ONE: " << search_for <<", "<< i->first << ", " << i->second << endl;
          //  exit(0);
          //}
          return i->second*multiplierSpVar;
      } else {
          //cout << "NOTM:  " << search_for <<", "<< i->first << endl;
          return 1.0*multiplierSpVar;
      }

    }
}
```

Here is the call graph for this function:



Here is the caller graph for this function:


ModelCoreSpatial::cachePixel ExogenousData
ModelCoreSpatial::runInit Period
getMultiplier
ModelCoreSpatial::compute CumulativeData
ModelCoreSpatial::runSimulationYear
Output::printDebugPixelValues

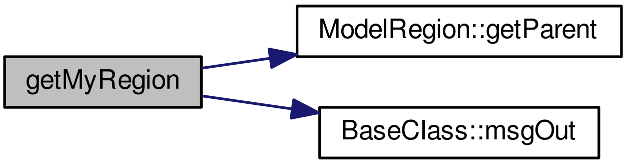
4.33.3.9 ModelRegion ∗ getMyRegion (const int & *rLevel* = 2)

Definition at line 355 of file Pixel.cpp.

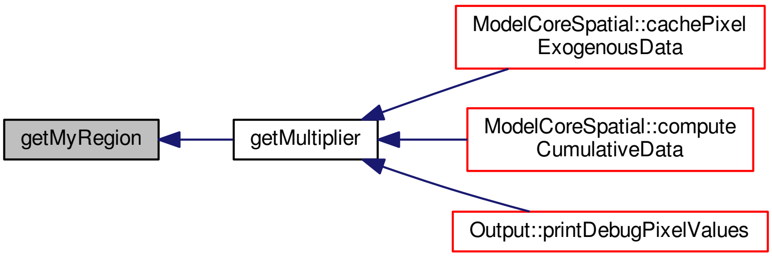
Referenced by getMultiplier().

```
00355                                       {
00356     if(rLevel==2){
00357         return l2region;
00358     } else if (rLevel==1) {
00359         return l2region->getParent();
00360     } else {
00361         msgOut(MSG_ERROR, "Requested a unknown level region code in getMyRegion().");
00362     }
00363 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


getMyRegion
getMultiplier
ModelCoreSpatial::cachePixel ExogenousData
ModelCoreSpatial::compute CumulativeData
Output::printDebugPixelValues

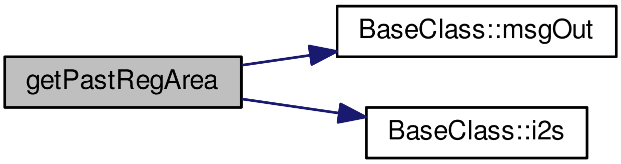
4.33.3.10 double getPastRegArea (const int & *ft_idx*, const int & *year*)

Definition at line 296 of file Pixel.cpp.

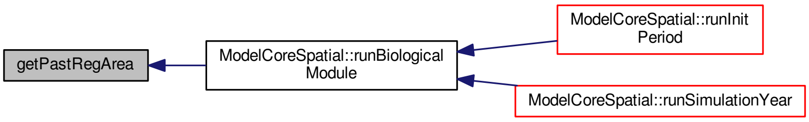
Referenced by ModelCoreSpatial::runBiologicalModule().

```
00296                                                         {
00297   map <int,vector<double> >::const_iterator i=regArea.find(year);
00298   if(i != regArea.end()) {
00299     return i->second.at(ft_idx);
00300   } else {
00301     msgOut(MSG_ERROR, "Asking for a pastRegArea of a not-registered year. I don't have year
      "+i2s(year)+"!");
00302   }
00303 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


getPastRegArea
ModelCoreSpatial::runBiological Module
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

4.33.3.11 double getPathMortality (const string & *forType*, const string & *dC*, int *year* = DATA_NOW)

Return the INCREASED mortality due to pathogen presence for a given ft and dc in a certain year (default the running year)

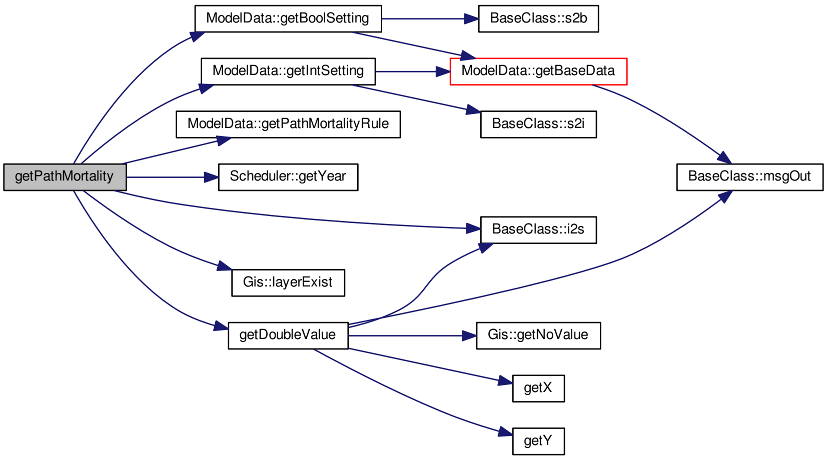
The mortality returned is the increased yearly mortality due to any affecting pathogenes. The function load the relevant pathogen mortality rule(s), for each of them check for how many years the phatogen is present with concentrations above the threshold and returns the relavant increase in mortality (summing them in case of multiple pathogens).

Definition at line 235 of file Pixel.cpp.

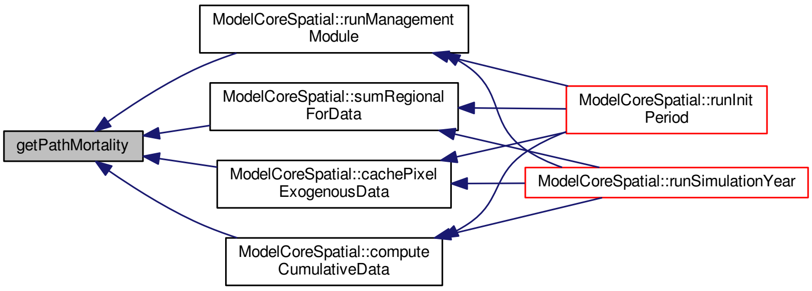
Referenced by ModelCoreSpatial::cachePixelExogenousData(), ModelCoreSpatial::computeCumulativeData(), ModelCoreSpatial::runManagementModule(), and ModelCoreSpatial::sumRegionalForData().

```
{
  if(!MTHREAD->MD->getBoolSetting("usePathogenModule")) return 0.0;

  string debug=forType;
  int initialOptYear = MTHREAD->MD->getIntSetting("initialOptYear");
  int simulationYears = MTHREAD->MD->getIntSetting("simulationYears");

  int maxYear = initialOptYear + simulationYears;

  vector<pathRule*> pathRules = MTHREAD->MD->getPathMortalityRule(
forType,dC);

  double pathMort = 0.0;
  if(year==DATA_NOW){year = MTHREAD->SCD->getYear();}

  for(uint r=0;r<pathRules.size();r++){
    string pathId=pathRules[r]->pathId;
    double pres_min=pathRules[r]->pres_min;
    vector<double> mortCoefficients=pathRules[r]->mortCoefficents;
    double pathMort_thispath = 0.0;
    for(uint y=year;y>(year-mortCoefficients.size());y--){
      int i =year-y;
      int y2 = y;
      if(y>=maxYear){
        y2=maxYear-1;
      }

      string layerName="pathogen_pp#"+pathId+"#"+i2s(y2);
      if(MTHREAD->GIS->layerExist(layerName)){
        if (this->getDoubleValue(layerName,true)>= pres_min){
          pathMort_thispath = mortCoefficients[i];
        }
      }
    }
    pathMort += pathMort_thispath;
  }
  return pathMort;

}
```

Here is the call graph for this function:



Here is the caller graph for this function:


getPathMortality
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::sumRegional ForData
ModelCoreSpatial::cachePixel ExogenousData
ModelCoreSpatial::compute CumulativeData
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

4.33.3.12 vector< Pixel * > getPixelsAtDistLevel (int *distLevel_h*) const

Return a vector of pixels at the specified distance (in levels, not in physical units)

The function return a vector of pointers to Pixels at the gived distance from the caller pixel.\ The list start with those on the Top, then add those on the right, those on the bottom and those on the left. Finally it had the corner pixels (that are more far).\ It takes into consideration borders correctly.

Fully tested on internal points as well semi-border cases, border cases and corner cases. ALL OK.

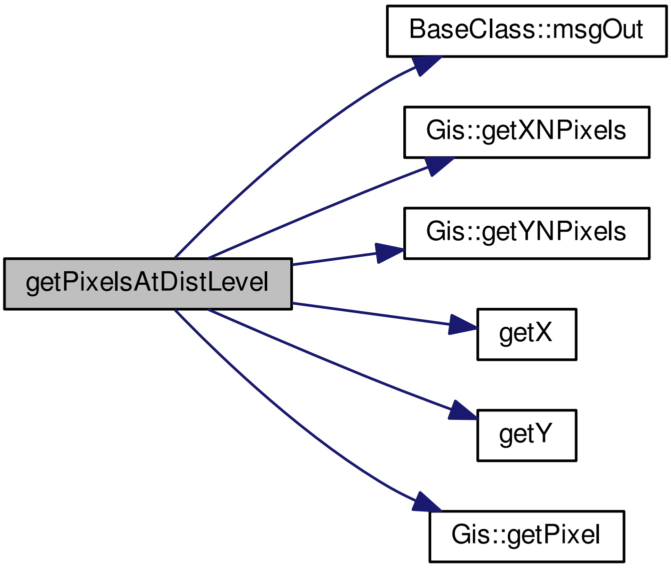
Parameters

distLevel_h	Distance in number of adiacent pixels. It has to be at least 1 (the function return an error if it is 0).

Definition at line 53 of file Pixel.cpp.

```
00053                                                {
00054
00055   if (distLevel_h<1) {
00056     msgOut(MSG_CRITICAL_ERROR, "getPixelsAtDistLevel() is defined for distances of
       at least 1 !");
00057   }
00058
00059   vector <Pixel *> toReturn;
00060   int xNPixels = MTHREAD->GIS->getXNPixels();
00061   int yNPixels = MTHREAD->GIS->getYNPixels();
00062   int thisX = this->getX();
00063   int thisY = this->getY();
00064   int minX = max(0        , (thisX - distLevel_h)+1);
00065   int maxX = min(xNPixels , thisX + distLevel_h);
00066   int minY = max(0        , (thisY - distLevel_h)+1);
00067   int maxY = min(yNPixels , thisY + distLevel_h);
00068
00069   // getting the top pixels (corner exluded)...
00070   if (thisY-distLevel_h >=0){
00071     for(int i=minX;i<maxX;i++){
00072       toReturn.push_back(MTHREAD->GIS->getPixel(i,thisY-distLevel_h));
00073     }
00074   }
00075   // getting the right pixels (corner exluded)...
00076   if (thisX+distLevel_h < xNPixels){
00077     for(int i=minY;i<maxY;i++){
00078       toReturn.push_back(MTHREAD->GIS->getPixel(thisX+distLevel_h,i));
00079     }
00080   }
00081   // getting the bottom pixels (corner exluded)...
00082   if (thisY+distLevel_h < yNPixels){
    for(int i=minX;i<maxX;i++){
      toReturn.push_back(MTHREAD->GIS->getPixel(i,thisY+distLevel_h));
    }
  }
  // getting the left pixels (corner exluded)...
  if (thisX-distLevel_h >= 0){
    for(int i=minY;i<maxY;i++){
      toReturn.push_back(MTHREAD->GIS->getPixel(thisX-distLevel_h,i));
    }
  }

  // getting the corners (correctly at the end, after already retrieved the other pixels..)...
  // top-left..
  if (thisX-distLevel_h >= 0 && thisY-distLevel_h >=0){
    toReturn.push_back(MTHREAD->GIS->getPixel(thisX-distLevel_h,thisY-distLevel_h));
  }
  // top-right..
  if (thisX+distLevel_h < xNPixels && thisY-distLevel_h >=0){
    toReturn.push_back(MTHREAD->GIS->getPixel(thisX+distLevel_h,thisY-distLevel_h));
  }
  // bottom-right..
  if (thisX+distLevel_h < xNPixels && thisY+distLevel_h <yNPixels){ // bug discovered 20070719
    toReturn.push_back(MTHREAD->GIS->getPixel(thisX+distLevel_h,thisY+distLevel_h));
  }
  // bottom-left..
  if (thisX-distLevel_h >= 0 && thisY+distLevel_h <yNPixels){
    toReturn.push_back(MTHREAD->GIS->getPixel(thisX-distLevel_h,thisY+distLevel_h));
  }
  return toReturn;
}
```

Here is the call graph for this function:



4.33.3.13 string getPxComments ( ) const [inline]

Definition at line 72 of file Pixel.h.

```
00072 {return pxComments;};
```

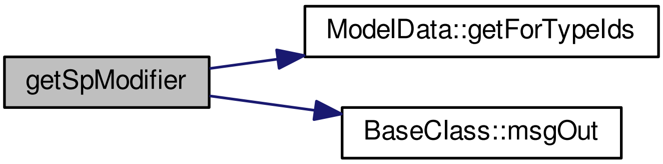
4.33.3.14 double getSpModifier (const string & *ft*)

Definition at line 343 of file Pixel.cpp.

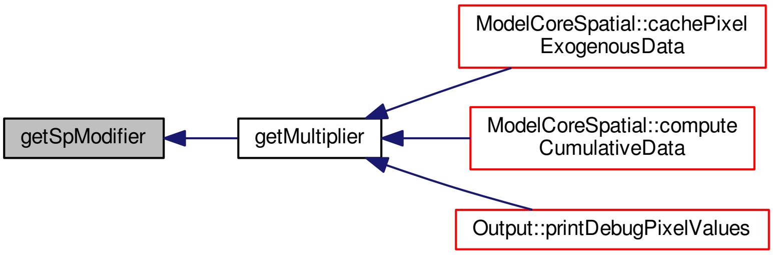
Referenced by getMultiplier().

```
00343                                     {
00344   vector<string>ftypes = MTHREAD->MD->getForTypeIds();
00345   for (int i=0;i<ftypes.size();i++){
00346       if (ftypes[i] == ft){
00347           return spMods.at(i);
00348       }
00349   }
00350   msgOut(MSG_CRITICAL_ERROR,"Asked spatial modifier for a forest type that doesn't
       exist");
00351 
00352 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


getSpModifier
getMultiplier
ModelCoreSpatial::cachePixel ExogenousData
ModelCoreSpatial::compute CumulativeData
Output::printDebugPixelValues

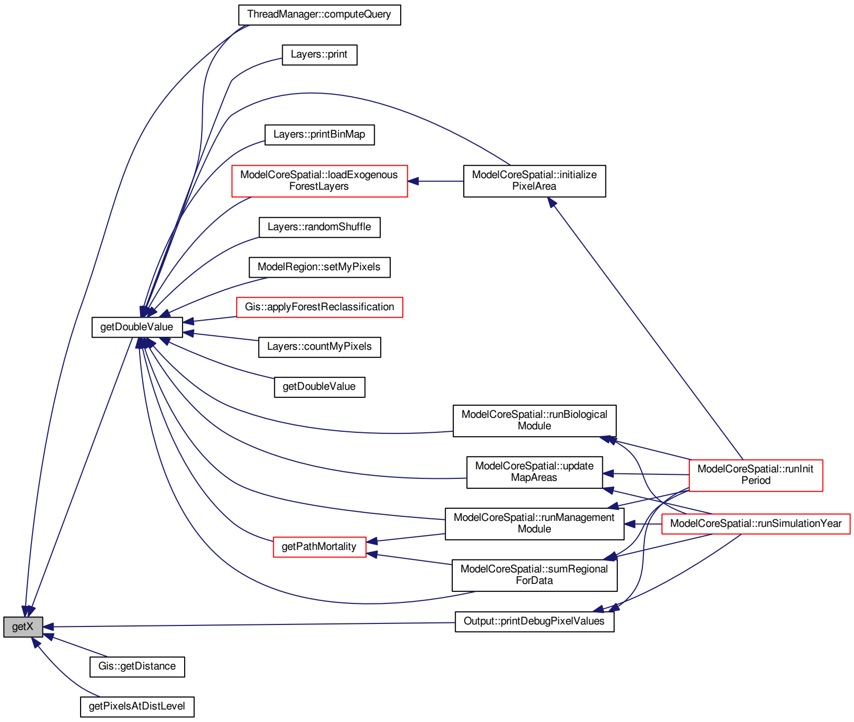
4.33.3.15 int getX ( ) const [inline]

Definition at line 67 of file Pixel.h.

Referenced by ThreadManager::computeQuery(), Gis::getDistance(), getDoubleValue(), getPixelsAtDistLevel(), and Output::printDebugPixelValues().

```
00067 {return pxX;} ;
```

Here is the caller graph for this function:


ThreadManager::computeQuery
Layers::print
Layers::printBinMap
ModelCoreSpatial::loadExogenous ForestLayers
ModelCoreSpatial::initialize PixelArea
Layers::randomShuffle
ModelRegion::setMyPixels
Gis::applyForestReclassification
getDoubleValue
Layers::countMyPixels
getDoubleValue
ModelCoreSpatial::runBiological Module
ModelCoreSpatial::update MapAreas
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::runSimulationYear
getPathMortality
ModelCoreSpatial::sumRegional ForData
getX
Output::printDebugPixelValues
Gis::getDistance
getPixelsAtDistLevel

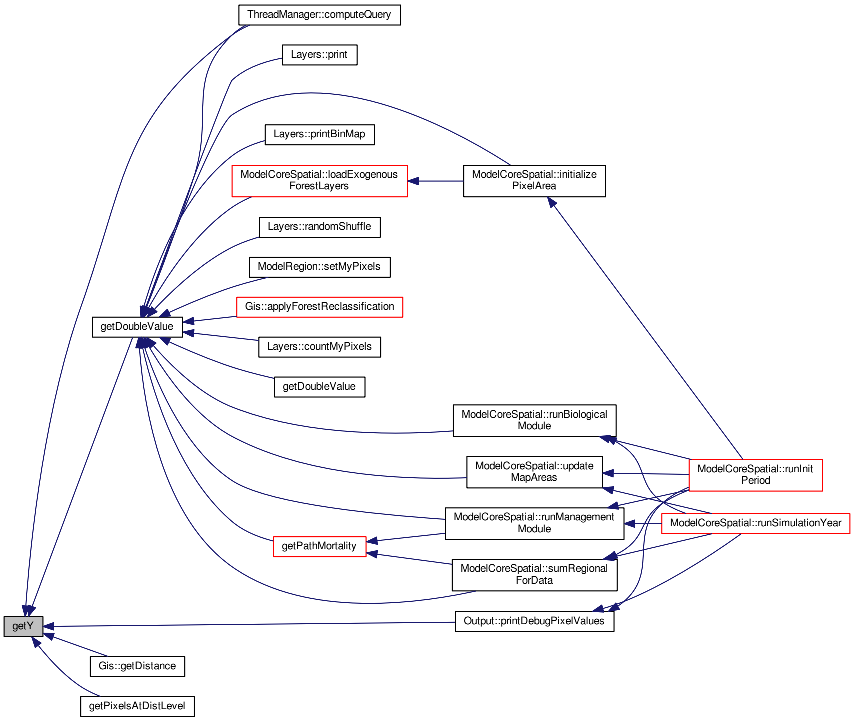
4.33.3.16 int getY ( ) const [inline]

Definition at line 68 of file Pixel.h.

Referenced by ThreadManager::computeQuery(), Gis::getDistance(), getDoubleValue(), getPixelsAtDistLevel(), and Output::printDebugPixelValues().

```
00068 {return pxY;};
```

Here is the caller graph for this function:


ThreadManager::computeQuery
Layers::print
Layers::printBinMap
ModelCoreSpatial::loadExogenous ForestLayers
ModelCoreSpatial::initialize PixelArea
Layers::randomShuffle
ModelRegion::setMyPixels
Gis::applyForestReclassification
getDoubleValue
Layers::countMyPixels
getDoubleValue
ModelCoreSpatial::runBiological Module
ModelCoreSpatial::update MapAreas
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::runSimulationYear
getPathMortality
ModelCoreSpatial::sumRegional ForData
getY
Output::printDebugPixelValues
Gis::getDistance
getPixelsAtDistLevel

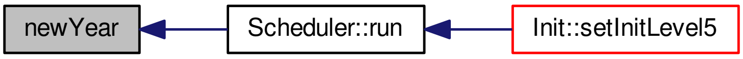
4.33.3.17 void newYear ( )

Definition at line 291 of file Pixel.cpp.

Referenced by Scheduler::run().

```
00291                {
00292
00293 }
```

Here is the caller graph for this function:



4.33.3.18 void setCachedDouble (double *cachedDouble_h*) [inline]

Definition at line 84 of file Pixel.h.

```
00084 {cachedDouble=cachedDouble_h;};
```

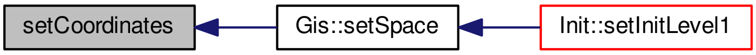
4.33.3.19 void setCoordinates (int *x_h,* int *y_h*) [inline]

Definition at line 82 of file Pixel.h.

Referenced by Gis::setSpace().

```
00082 {pxX=x_h; pxY=y_h;};
```

Here is the caller graph for this function:



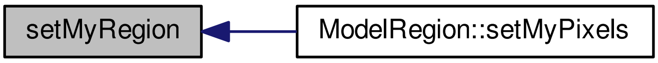
4.33.3.20 void setMyRegion (ModelRegion ∗ *region_h*) [inline]

Definition at line 89 of file Pixel.h.

Referenced by ModelRegion::setMyPixels().

```
00089 {l2region = region_h;};
```

Here is the caller graph for this function:

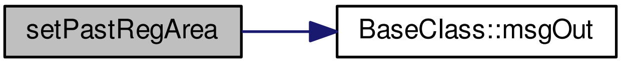


4.33.3.21 void setPastRegArea (const double & *value,* const int & *ft_idx,* const int & *year*)

Definition at line 306 of file Pixel.cpp.

```
00306                                                                           {
00307   msgOut(MSG_CRITICAL_ERROR,"TODO");
00308   /*map <int,vector<double> >::const_iterator i=regArea.find(year);
00309   if(i != regArea.end()) {
00310     // we already have this year, let's see if the vector is bif enough
00311     int currsize = i->second.size();
00312     for(j=0;j<ft_idx-currside;j++){
00313 
00314     }
00315     return i->second.at(ft_idx);
00316   } else {
00317     // new year
00318   }
00319 
00320 
00321   pair<int,vector<double> newRegArea;
00322   */
00323 
00324 
00325 }
```

Here is the call graph for this function:



4.33.3.22 void setPxComments (std::string *pxComments_h*) [inline]

Definition at line 83 of file Pixel.h.

```
00083 {pxComments = pxComments_h;};
```

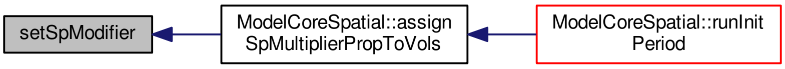
4.33.3.23 void setSpModifier (const double & *value,* const int & *ftindex*) [inline]

Definition at line 86 of file Pixel.h.

Referenced by ModelCoreSpatial::assignSpMultiplierPropToVols().

```
00086 {spMods.at(ftindex)=value;};
```

Here is the caller graph for this function:


setSpModifier
ModelCoreSpatial::assign SpMultiplierPropToVols
ModelCoreSpatial::runInit Period

4.33.3.24 void setValue (const string & *layerName_h,* const double & *value_h*) [inline]

Insert a new layer and its value.

Definition at line 77 of file Pixel.h.

```
00077 {values.insert(pair<string, double>(layerName_h, value_h));}
```

4.33.3.25 void swap (const int & *swap_what*)

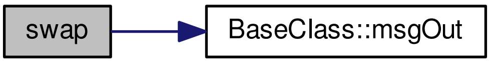
Assign to the delayed value the current values, e.g. vol_l = vol.

Definition at line 328 of file Pixel.cpp.

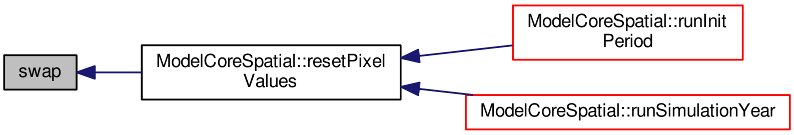
Referenced by ModelCoreSpatial::resetPixelValues().

```
00328                                 {
00329   switch (swap_what){
00330     case VAR_VOL:
00331       vol_l = vol;
00332       break;
00333     case VAR_AREA:
00334       area_l = area;
00335       break;
00336     default:
00337       msgOut(MSG_CRITICAL_ERROR,"Don't know how to swap "+swap_what);
00338   }
00339 }
```

Here is the call graph for this function:



Here is the caller graph for this function:


swap
ModelCoreSpatial::resetPixel Values
ModelCoreSpatial::runInit Period
ModelCoreSpatial::runSimulationYear

4.33.4 Member Data Documentation

4.33.4.1 vector<vector <double> > area

Definition at line 106 of file Pixel.h.

Referenced by ModelCoreSpatial::initializePixelArea(), ModelCoreSpatial::loadExogenousForestLayers(), Output::printDebugPixelValues(), ModelCoreSpatial::resetPixelValues(), ModelCoreSpatial::runBiologicalModule(), ModelCoreSpatial::runManagementModule(), ModelCoreSpatial::sumRegionalForData(), swap(), and ModelCoreSpatial::updateMapAreas().

4.33.4.2 vector< vector < double > > area_l

store the areas of the previous year

Definition at line 120 of file Pixel.h.

Referenced by ModelCoreSpatial::initializePixelArea(), ModelCoreSpatial::loadExogenousForestLayers(), ModelCoreSpatial::resetPixelValues(), ModelCoreSpatial::runBiologicalModule(), and swap().

4.33.4.3 double avalCoef

Availability (of wood resources) coefficient. A [0,1] coefficient that reduce avaiability of wood resources to exploitation due to local reasons (protected area, altimetry..)

Definition at line 137 of file Pixel.h.

Referenced by ModelCoreSpatial::computeInventary(), ModelData::getAvailableAliveTimber(), Pixel(), and ModelCoreSpatial::runBiologicalModule().

4.33.4.4 vector< vector < double > > beta

Definition at line 122 of file Pixel.h.

Referenced by ModelCoreSpatial::cachePixelExogenousData(), ModelCoreSpatial::resetPixelValues(), and ModelCoreSpatial::runBiologicalModule().

4.33.4.5 double cachedDouble [private]

Cachable double used in some optimized algorithms.

Definition at line 146 of file Pixel.h.

4.33.4.6 vector< vector < double > > cumAlive

Cumulative prob of remaining alive at beginnin of a given diam class.

Definition at line 127 of file Pixel.h.

Referenced by ModelCoreSpatial::computeCumulativeData(), and ModelCoreSpatial::resetPixelValues().

4.33.4.7 vector< vector < double > > cumAlive_exp

This is the **expected** version of cumAlive, used for calculating profits.

Definition at line 130 of file Pixel.h.

Referenced by ModelCoreSpatial::computeCumulativeData(), ModelCoreSpatial::resetPixelValues(), and ModelCoreSpatial::runManagementModule().

4.33.4.8 vector< vector < double > > cumTp

This is time of passage to REACH a diameter class (while the exogenous tp by diameter class is the time of passage to LEAVE to the next d class)

Definition at line 125 of file Pixel.h.

Referenced by ModelCoreSpatial::computeCumulativeData(), and ModelCoreSpatial::resetPixelValues().

4.33.4.9 vector< vector < double > > cumTp_exp

This is the **expected** version of cumTp, used for calculating profits.

Definition at line 128 of file Pixel.h.

Referenced by ModelCoreSpatial::computeCumulativeData(), Output::printDebugPixelValues(), ModelCoreSpatial::resetPixelValues(), and ModelCoreSpatial::runManagementModule().

4.33.4.10 vector< double > expectedReturns

Definition at line 116 of file Pixel.h.

Referenced by Output::printDebugPixelValues(), ModelCoreSpatial::resetPixelValues(), ModelCoreSpatial::runManagementModule(), ModelCoreSpatial::sumRegionalForData(), and ModelCoreSpatial::updateOtherMapData().

4.33.4.11 vector< double > expectedReturnsNotCorrByRa

by ft. Attenction, reported expReturns at "forest" level (compared with those at forest type level) do NOT include ra

Definition at line 117 of file Pixel.h.

Referenced by ModelCoreSpatial::runManagementModule(), and ModelCoreSpatial::sumRegionalForData().

4.33.4.12 double expType

Sampling derived expectation types of this agent (forest bilogical parameters: growth, mortality)

Definition at line 134 of file Pixel.h.

Referenced by ModelCoreSpatial::computeCumulativeData().

4.33.4.13 double expTypePrices

Sampling derived expectation types of this agent (prices)

Definition at line 135 of file Pixel.h.

Referenced by ModelCoreSpatial::runManagementModule().

4.33.4.14 vector< vector < double > > hArea

Definition at line 108 of file Pixel.h.

Referenced by ModelCoreSpatial::resetPixelValues(), ModelCoreSpatial::runBiologicalModule(), ModelCoreSpatial::runManagementModule(), ModelCoreSpatial::sumRegionalForData(), and ModelCoreSpatial::updateMapAreas().

4.33.4.15 vector< vector < double > > hVol

Definition at line 109 of file Pixel.h.

Referenced by ModelCoreSpatial::resetPixelValues(), ModelCoreSpatial::runBiologicalModule(), and ModelCoreSpatial::sumRegionalForData().

4.33.4.16 vector< vector <vector <double> > > hVol_byPrd

Definition at line 110 of file Pixel.h.

Referenced by ModelCoreSpatial::resetPixelValues(), and ModelCoreSpatial::runBiologicalModule().

4.33.4.17 double ID `[private]`

Definition at line 142 of file Pixel.h.

4.33.4.18 vector<double> initialDc0Area

Definition at line 107 of file Pixel.h.

Referenced by ModelCoreSpatial::runBiologicalModule().

4.33.4.19 ModelRegion* l2region `[private]`

Pointer to level 2 region where this pixel is.

Definition at line 148 of file Pixel.h.

Referenced by getMyRegion().

4.33.4.20 vector<vector <double> > mort

Definition at line 123 of file Pixel.h.

Referenced by ModelCoreSpatial::cachePixelExogenousData(), ModelCoreSpatial::resetPixelValues(), and ModelCoreSpatial::runBiologicalModule().

4.33.4.21 double portfolioVarRa

Sampling derived risk aversion on portfolio variance for of this agent.

Definition at line 133 of file Pixel.h.

4.33.4.22 string pxComments `[private]`

Definition at line 145 of file Pixel.h.

4.33.4.23 int pxX `[private]`

Definition at line 143 of file Pixel.h.

4.33.4.24 int pxY `[private]`

Definition at line 144 of file Pixel.h.

4.33.4.25 map<int, vector <double> > regArea

Definition at line 111 of file Pixel.h.

Referenced by getPastRegArea(), ModelCoreSpatial::runManagementModule(), ModelCoreSpatial::sumRegionalForData(), and ModelCoreSpatial::updateMapAreas().

4.33.4.26 vector<double> spMods [private]

The sampled spatial modifiers (by forest type)

Definition at line 147 of file Pixel.h.

Referenced by getSpModifier(), and Pixel().

4.33.4.27 vector<vector <double> > tp

Definition at line 124 of file Pixel.h.

Referenced by ModelCoreSpatial::cachePixelExogenousData(), ModelCoreSpatial::initializePixelArea(), ModelCoreSpatial::resetPixelValues(), and ModelCoreSpatial::runBiologicalModule().

4.33.4.28 bool usePortfolio

Sampling derived usage of portfolio management (false/true)

Definition at line 136 of file Pixel.h.

4.33.4.29 map<string, double> values [private]

Map of values for each layer.

Definition at line 140 of file Pixel.h.

Referenced by changeValue(), correctInputMultiplier(), getDoubleValue(), and getMultiplier().

4.33.4.30 vector<vector <double> > vHa

Volume at hectar by each diameter class [m^3/ha].

Definition at line 126 of file Pixel.h.

Referenced by ModelCoreSpatial::computeCumulativeData(), ModelCoreSpatial::initializePixelArea(), ModelCoreSpatial::resetPixelValues(), and ModelCoreSpatial::runBiologicalModule().

4.33.4.31 vector<vector <double> > vHa_exp

This is the **expected** version of vHa, used for calculating profits.

Definition at line 129 of file Pixel.h.

Referenced by ModelCoreSpatial::computeCumulativeData(), Output::printDebugPixelValues(), ModelCoreSpatial::resetPixelValues(), and ModelCoreSpatial::runManagementModule().

4.33.4.32 map<string, double>::const_iterator vIter [mutable], [private]

Definition at line 141 of file Pixel.h.

Referenced by getDoubleValue().

4.33.4.33 vector<vector <double> > vMort

Definition at line 115 of file Pixel.h.

Referenced by ModelCoreSpatial::resetPixelValues(), ModelCoreSpatial::runBiologicalModule(), and ModelCoreSpatial::sumRegionalForData().

4.33.4.34 vector<vector <double> > vol

Definition at line 89 of file Pixel.h.

Referenced by ModelCoreSpatial::assignSpMultiplierPropToVols(), ModelCoreSpatial::loadExogenousForestLayers(), Output::printDebugPixelValues(), ModelCoreSpatial::resetPixelValues(), ModelCoreSpatial::runBiologicalModule(), ModelCoreSpatial::sumRegionalForData(), swap(), and ModelCoreSpatial::updateOtherMapData().

4.33.4.35 vector<vector <double> > vol_l

store the volumes of the previous year

Definition at line 119 of file Pixel.h.

Referenced by ModelCoreSpatial::computeInventary(), ModelData::getAvailableAliveTimber(), ModelCoreSpatial::initializePixelArea(), ModelCoreSpatial::runBiologicalModule(), and swap().

4.33.4.36 vector<double> vReg

Definition at line 114 of file Pixel.h.

Referenced by ModelCoreSpatial::resetPixelValues(), ModelCoreSpatial::runBiologicalModule(), and ModelCoreSpatial::sumRegionalForData().

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

- /home/lobianco/git/ffsm_pp/src/Pixel.h
- /home/lobianco/git/ffsm_pp/src/Pixel.cpp

4.34 ReclassRules Struct Reference

Initial reclassification rules (dataset filters)

`#include <Layers.h>`

Public Attributes

- int inCode
- int outCode
- double p

  Probability that one pixel of code inCode will become of code outCode. 1 for fixed transformation.

4.34.1 Detailed Description

Initial reclassification rules (dataset filters)

A structure for easy reclassification of "mixed" categories in some layers.
The reclassification can be made to both *increase* depth or *decrease* depth to the original dataset.
Eg, if in our model we don't differ between coniferous and hardwood forests, we can set all them to be "forest".
At the opposite, if our model require more detail than the map provide, e.g. irrigable arable VS dry arable, we can set the generic "arable land" of becoming "arable" or "dry" according with a regional-defined probability (getted from other sources, e.g. census data).

Author

Antonello Lobianco

Definition at line 135 of file Layers.h.

4.34.2 Member Data Documentation

4.34.2.1 int inCode

Definition at line 136 of file Layers.h.

4.34.2.2 int outCode

Definition at line 137 of file Layers.h.

4.34.2.3 double p

Probability that one pixel of code inCode will become of code outCode. 1 for fixed transformation.

Definition at line 139 of file Layers.h.

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

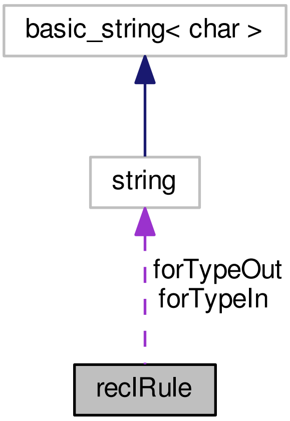
- /home/lobianco/git/ffsm_pp/src/Layers.h

4.35 reclRule Struct Reference

IO production matrix between the forest resources and the primary products (struct)

`#include <ModelData.h>`

Collaboration diagram for reclRule:



Public Attributes

- int regId
- string forTypeIn
- string forTypeOut
- double coeff

4.35.1 Detailed Description

IO production matrix between the forest resources and the primary products (struct)

Struct containing the io matrix between the forest resources and the primary products. Not to be confunded with the IO matrix between primary products and secondary products.

Definition at line 297 of file ModelData.h.

4.35.2 Member Data Documentation

4.35.2.1 double coeff

Definition at line 301 of file ModelData.h.

Referenced by Gis::applyForestReclassification(), ModelData::applyOverrides(), and ModelData::setReclassification↩ Rules().

4.35.2.2 string forTypeIn

Definition at line 299 of file ModelData.h.

Referenced by Gis::applyForestReclassification(), ModelData::applyOverrides(), and ModelData::setReclassification↩ Rules().

4.35.2.3 string forTypeOut

Definition at line 300 of file ModelData.h.

Referenced by Gis::applyForestReclassification(), ModelData::applyOverrides(), and ModelData::setReclassification↩ Rules().

4.35.2.4 int regId

Definition at line 298 of file ModelData.h.

Referenced by Gis::applyForestReclassification(), ModelData::applyOverrides(), and ModelData::setReclassification↩ Rules().

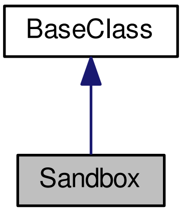
The documentation for this struct was generated from the following file:

- /home/lobianco/git/ffsm_pp/src/ModelData.h

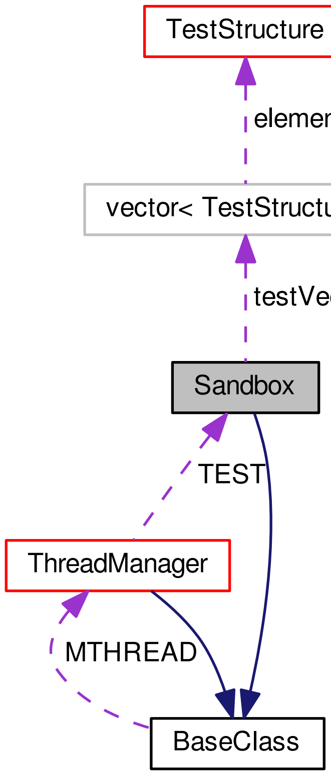
4.36 Sandbox Class Reference

`#include <Sandbox.h>`

Inheritance diagram for Sandbox:



Collaboration diagram for Sandbox:



Public Member Functions

- Sandbox (ThreadManager ∗MTHREAD_h)
- Sandbox ()
- ∼Sandbox ()
- template< class T >
  T getSetting (string name_h, int type)
- template< class T >
  vector< T > getVectorSetting (string name_h, int type)
- template< class T >
  T test2 (const std::string &s)
- void printAString (string what)
- vector< TestStructure ∗ > getTestStructure ()
- void testThreads ()
- void basicTest ()

  Simple tests that doesn't require anything else (are encapsulated) and so they can be run at the beginning of the program. Normally empty.
- void fullTest ()

  Tests that require a full sandbox object including MTHREAD. Normally empty.
- void testIpopt ()
- int testAdolc ()

- void testPartMatching ()

  How to partial matching the key of a map.
- void testPartMatching2 ()

  How to partial matching the key of a map.

Private Member Functions

- void testSearchMap (const map< string, string > &map, const string &search_for)
- void testSearchMap2 (const map< string, string > &map_h, const string &search_for)

Private Attributes

- vector< TestStructure > testVector

Additional Inherited Members

4.36.1 Detailed Description

Definition at line 40 of file Sandbox.h.

4.36.2 Constructor & Destructor Documentation

4.36.2.1 Sandbox (ThreadManager ∗ *MTHREAD_h*)

Definition at line 80 of file Sandbox.cpp.

```
00080                                       {
00081   MTHREAD=MTHREAD_h;
00082 }
```

4.36.2.2 Sandbox ( )

Definition at line 84 of file Sandbox.cpp.

```
00084                  {
00085
00086 }
```

4.36.2.3 ∼Sandbox ( )

Definition at line 89 of file Sandbox.cpp.

```
00089                   {
00090
00091 }
```

4.36.3 Member Function Documentation

4.36.3.1 void basicTest ( )

Simple tests that doesn't require anything else (are encapsulated) and so they can be run at the beginning of the program. Normally empty.

Definition at line 131 of file Sandbox.cpp.

Referenced by main(), and printAString().

```
00131                   {
00132 
00133     /*
00134     // Testing debugging a map
00135     iisskey k1(2007,11021,"broadL_HighF","15");
00136     iisskey k2(2007,11021,"broadL_HighF","30");
00137     iisskey k3(2007,11021,"con_HighF","15");
00138     iisskey k4(2007,11022,"broadL_HighF","15");
00139     iisskey k5(2008,11021,"broadL_HighF","15");
00140 
00141     // Testing the new changeMapValue(), incrMapValue(), resetMapValues(), incrOrAddMapValue(map, key,
       value) and vectorToMap() funcions
00142     map<iisskey,double> testMap;
00143     pair<iisskey,double> pair1(k1,1.1);
00144     pair<iisskey,double> pair2(k2,1.2);
00145     pair<iisskey,double> pair3(k3,1.3);
00146     pair<iisskey,double> pair4(k4,1.4);
00147     pair<iisskey,double> pair5(k5,1.5);
00148     testMap.insert(pair1);
00149     testMap.insert(pair2);
00150     testMap.insert(pair3);
00151     testMap.insert(pair4);
00152     testMap.insert(pair5);
00153     debugMap(testMap,iisskey(NULL,NULL,"",""));
00154     debugMap(testMap,iisskey(2007,NULL,"con_HighF",""));
00155     exit(0);
00156     */
00157 
00158 
00159 
00160 
00161   /*
00162   // Testing standard deviation algorithm, as from http://stackoverflow.com/questions/7616511/
      calculate-mean-and-standard-deviation-from-a-vector-of-samples-in-c-using-boos
00163   vector<double> v;
00164   v.push_back(3.0);
00165   v.push_back(2.0);
00166   v.push_back(5.0);
00167   v.push_back(4.0);
00168   double sum = std::accumulate(std::begin(v), std::end(v), 0.0);
00169   double m =  sum / v.size();
00170   double accum = 0.0;
00171   std::for_each (std::begin(v), std::end(v), [&](const double d) {
00172       accum += (d - m) * (d - m);
00173   });
00174   double stdev = sqrt(accum / (v.size()-1));
00175   cout << stdev << endl;
00176   double sd2 = getSd(v);
00177   double sd3 = getSd(v,false);
00178   cout << sd2 << endl;
00179   cout << sd3 << endl;
00180   exit(0);
00181   */
00182 
00183   /*
00184   // Testing tokenize, untokenize functions
00185   vector<string> istrings;
00186   istrings.push_back("Questo");
00187   istrings.push_back("cielo");
00188   istrings.push_back("è");
00189   istrings.push_back("sempre");
00190   istrings.push_back("più");
00191   istrings.push_back("blu.");
00192   string delimiter = " . ";
00193 
00194   string fullstring="";
00195   vector<string> ostrings;
00196   untokenize(fullstring, istrings, delimiter);
00197   cout << fullstring << endl;
```

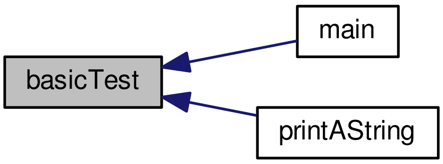
```

  fullstring += delimiter;
  cout << fullstring << endl;

  tokenize(fullstring, ostrings, delimiter);
  for (uint i=0;i<ostrings.size();i++){
    cout << ostrings[i] << endl;
  }
  exit(0);
  */


  /*
  // Testing FlopC++
  // For a single file compile as:
  // -- two passages:
  // g++ -O3 -I /usr/include/coin -DFLOPCPP_BUILD `PKG_CONFIG_PATH=/usr/lib64/pkgconfig:/usr/lib/pkgconfig:/usr/share/pkgconfig: pkg-config --libs flopcpp osi-cbc osi-clp` transport.cpp -c -o  transport.o
  // g++ -o transport2 transport.o -Wl,-rpath,'$ORIGIN' -L . -DFLOPCPP_BUILD `PKG_CONFIG_PATH=/usr/lib64/pkgconfig:/usr/lib/pkgconfig:/usr/share/pkgconfig: pkg-config --libs flopcpp osi-cbc osi-clp`
  // -- single passage:
  // g++ -O3 -I /usr/include/coin transport.cpp -DFLOPCPP_BUILD `PKG_CONFIG_PATH=/usr/lib64/pkgconfig:/usr/lib/pkgconfig:/usr/share/pkgconfig: pkg-config --libs flopcpp osi-cbc osi-clp`  -o  transport3

  MP_model::getDefaultModel().setSolver(new OsiClpSolverInterface);
  //MP_model::getDefaultModel().setSolver(new OsiCbcSolverInterface);
  enum  {seattle, sandiego, numS};
  enum  {newyork, chicago, topeka,numD};

  MP_set S(numS);          // Sources
  MP_set D(numD);          // Destinations
  MP_subset<2> Link(S,D);  // Transportation links (sparse subset of S*D)

  Link.insert(seattle,newyork);
  Link.insert(seattle,chicago);
  Link.insert(sandiego,chicago);
  Link.insert(sandiego,topeka);

  MP_data SUPPLY(S);
  MP_data DEMAND(D);

  SUPPLY(seattle)=350;  SUPPLY(sandiego)=600;
  DEMAND(newyork)=325;  DEMAND(chicago)=300;  DEMAND(topeka)=275;

  MP_data COST(Link);

  COST(Link(seattle,newyork)) = 2.5;
  COST(Link(seattle,chicago)) = 1.7;
  COST(Link(sandiego,chicago))= 1.8;
  COST(Link(sandiego,topeka)) = 1.4;

  COST(Link) = 90 * COST(Link) / 1000.0;

  MP_variable x(Link);
  x.display("...");

  MP_constraint supply(S);
  MP_constraint demand(D);

  supply.display("...");

  supply(S) =  sum( Link(S,D), x(Link) ) <= SUPPLY(S);
  demand(D) =  sum( Link(S,D), x(Link) ) >= DEMAND(D);

  cout<<"Here"<<endl;

  minimize( sum(Link, COST(Link)*x(Link)) );
  assert(MP_model::getDefaultModel()->getNumRows()==5);
  assert(MP_model::getDefaultModel()->getNumCols()==4);
  assert(MP_model::getDefaultModel()->getNumElements()==8);
  assert(MP_model::getDefaultModel()->getObjValue()>=156.14 && MP_model::getDefaultModel()->getObjValue()<=156.16);

  x.display("Optimal solution:");
  supply.display("Supply dual solution");
  cout<<"Test transport passed."<<endl;
  */



  /*
  // Testing limits for 0
  double test = DBL_MIN;
  cout << test << endl;
  test = numeric_limits<double>::min();
  cout << test << endl;
  exit(0);
```

```
*/


/*
// Testing getMaxPos()
vector<double> test {7,2,6,4,7,2,5,7,2};
double maxpos   = getMaxPos(test);
double maxvalue = getMax(test);
double minpos   = getMinPos(test);
double minvalue = getMin(test);
//double maxpos = testB();
cout << "maxpos: " << maxpos << endl;
cout << "maxvalue: " << maxvalue << endl;
cout << "minpos: " << minpos << endl;
cout << "minvalue: " << minvalue << endl;
exit(0);
*/


/*
//This was in ModelData::debug():
// ********** START DEBUG CODE....... ************
double ddebuga=0; //20080209
uint idebuga=0;
double ddebugb=0; //20080209
uint idebugb=0;
double ddebugc=0; //20080209
uint idebugc=0;
double debugmin = 0;
double debugmax = 1000;
for (uint q=0;q<10000;q++){
  ddebuga += debugmin + ( (double)rand() / ((double)(RAND_MAX)+(double)(1)) )*(debugmax-debugmin+1);
  ddebugb += debugmin + ( (double)rand() / ((double)(RAND_MAX)+(double)(1)) )*(debugmax-debugmin+1);
  ddebugc += debugmin + ( (double)rand() / ((double)(RAND_MAX)+(double)(1)) )*(debugmax-debugmin+1);
}
idebuga = ddebuga;
idebugb = ddebugb;
idebugc = ddebugc;
cout << "idebuga: "<<idebuga<<endl;
cout << "idebugb: "<<idebugb<<endl;
cout << "idebugc: "<<idebugc<<endl;
throw 2;
// ******** .....END DEBUG CODE *******************
*/

  /*
  // Testing the new iskey class
  iskey op1(2100,"test");
  iskey op2(2100,"test");
  iskey op3(2101,"test");
  iskey op4(2101,"tgst");
  iskey op5(2101,"tb");
  iskey op6(2101,"testa");
  if(op1 == op2){
    cout << "op1 and op2 are equal" << endl;
  }
  if(op1 == op3){
    cout << "op1 and op3 are equal" << endl;
  }
  if(op6 > op3) cout << "test3 passed" << endl;
  if(op5 < op3) cout << "test4 passed" << endl;
  if(op6 >= op3) cout << "test5 passed" << endl;
  if(op6 != op3) cout << "test6 passed" << endl;
  if(op4 <= op3) cout << "test7 passed that it shoudn't" << endl;
  exit(0);
  */

  /*
  // Testing the new changeMapValue(), incrMapValue(), resetMapValues(), incrOrAddMapValue(map, key, value) and vectorToMap() funcions
  map<int,double> testMap;
  for (uint i=0;i<5;i++){
      pair<int,double> mypair(i,i*2.5);
      testMap.insert(mypair);
  }
  double result = findMap(testMap,3,MSG_NO_MSG);
  double result2 = findMap(testMap,1,MSG_ERROR);
  double result3 = findMap(testMap,7,MSG_DEBUG);
  cout << findMap(testMap,3,MSG_NO_MSG)<< endl;
  changeMapValue(testMap,3,200.0,MSG_ERROR);
  cout << findMap(testMap,3,MSG_NO_MSG)<< endl;
  incrMapValue(testMap,3,5.0,MSG_ERROR);
  cout << findMap(testMap,3,MSG_NO_MSG)<< endl;
  incrOrAddMapValue(testMap, 3, 200.0);
  cout << findMap(testMap,3,MSG_NO_MSG)<< endl;
  incrOrAddMapValue(testMap, 10, 100.0);
  cout << findMap(testMap,10,MSG_NO_MSG)<< endl;
```

```
   cout << "done" << endl;

   vector<string> mykeys;
   mykeys.push_back("andrea");
   mykeys.push_back("antonello");
   mykeys.push_back("paolo");
   map<string,double> mymap = vectorToMap(mykeys,15.0);
   string searchkey;
   searchkey = "andrea";
   cout << findMap(mymap,searchkey,MSG_DEBUG)<< endl;
   resetMapValues(mymap,32.0);
   cout << findMap(mymap,searchkey,MSG_DEBUG)<< endl;
   exit(0);
   */



   /*
   // -------------------------------------------------------------------
   // Sampling from uniform distribution with local random seed
   // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

   //this code sample from a uniform distribution. In this case also the seed is reinitialisated, but it it valid only locally: the rest of the program run with the same seed

   std::random_device rd;
   std::mt19937 gen(rd());
   std::uniform_int_distribution<> dis(1, 6);

   for (int n=0; n<10; ++n)
       std::cout << dis(gen) << ' ';
   std::cout << '\n';
   exit(0);
  */



 /*
 // -------------------------------------------------------------------
 // Testing how to get all elements in a map by substrings
 // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 map <string,double> values;
 pair <string,double> val1("AAAAAA",1);
 pair <string,double> val2("AAABBB",2);
 pair <string,double> val3("BBBAAA",3);
 pair <string,double> val4("BBBBBB",4);
 pair <string,double> val5("CCCAAA",5);
 pair <string,double> val6("C",6);
 pair <string,double> val7("BBB",7);

 values.insert(val1);
 values.insert(val2);
 values.insert(val3);
 values.insert(val4);
 values.insert(val5);
 values.insert(val6);
 values.insert(val7);

 cout << "Printing whole map" << endl;
 for (std::map<string,double>::iterator it=values.begin(); it!=values.end(); ++it)
     std::cout << it->first << " => " << it->second << '\n';

 string search_for = "BBB";

 cout << "Using lower bound " << endl;
 for (std::map<string,double>::iterator it=values.lower_bound(search_for); it!=values.end(); ++it)
     std::cout << it->first << " => " << it->second << '\n';
 cout << "Using upper bound " << endl;
 for (std::map<string,double>::iterator it=values.upper_bound(search_for); it!=values.end(); ++it)
     std::cout << it->first << " => " << it->second << '\n';

 cout << "Printing only substrings " << endl;
 for (std::map<string,double>::iterator it=values.lower_bound(search_for); it!=values.end(); ++it){
   string key = it->first;
   if (key.compare(0, search_for.size(), search_for) == 0){
     std::cout << it->first << " => " << it->second << '\n';
   }
 }


 exit(0);
 */

 /*
 // testing findMap
 map<int,double> testMap;
 for (uint i=0;i<5;i++){
   pair<int,double> mypair(i,i*2.5);
   testMap.insert(mypair);
 }
 double result = findMap(testMap,3,MSG_NO_MSG);
 double result2 = findMap(testMap,1,MSG_ERROR);
 double result3 = findMap(testMap,7,MSG_DEBUG);
 cout << "Done" << endl;
 map<int, vector <double> > testMap2;
 for (uint i=0;i<5;i++){
   vector <double> myvector;
   for(uint j=0;j<10;j++) {
     myvector.push_back(i*100+j);
   }
   pair<int,vector <double> > mypair2(i,myvector);
   testMap2.insert(mypair2);
 }
 vector <double> resultb = findMap(testMap2,3,MSG_NO_MSG);
 vector <double> resultb2 = findMap(testMap2,1,MSG_ERROR);
 vector <double> resultb3 = findMap(testMap2,7);
 cout << "Done2" << endl;
 exit(1);
 */



 /*
 // Testing vSum
 vector <int> ivector(5,5);
 vector <double> dvector(5,1.5);
 vector < vector <int> > ivector2;
 vector <vector <double > > dvector2;


 for(uint i=0;i<5;i++){
   ivector2.push_back(ivector);
   dvector2.push_back(dvector);
 }

 int iSum = vSum(ivector);
 int iSum2 = vSum(ivector2[2]);
 double dSum = vSum(dvector);
 double dSum2 = vSum(dvector2[1]);
 int iSum3 = vSum(ivector2);
 double dSum3 = vSum(dvector2);

 cout << "hi there" << endl;
 */

 /*
 // Testing Eigen
 using Eigen::MatrixXd;
 MatrixXd m(2,2);
 m(0,0) = 4;
 m(1,0) = 2.5;
 m(0,1) = -1;
 m(1,1) = m(1,0) + m(0,1);
 std::cout << m << std::endl;
 exit(0);
 */

 /*
 // Test on two different type of partial matching over map values
 testPartMatching2();
 testPartMatching();
 */

 /*
 // -------------------------------------------------------------------
 // Testing how to erase elements from a vector according to conditions
 // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

 vector<string> myvector;
 myvector.push_back("a");
 myvector.push_back("b");
 myvector.push_back("c");
 myvector.push_back("d");
 myvector.push_back("e");

 for (uint i=0; i<myvector.size();i++){
   cout << "i:" << i << " myvector[i]: " << myvector[i] << endl;
   if(myvector[i]== "c" || myvector[i]=="d"){
     cout << " -- TBR: " << "i:" << i << " myvector[i]: " << myvector[i] << endl;
     myvector.erase (myvector.begin()+i);
     i--;
   }
 }

```

```
  cout << "Myvector now contains:" << endl;
  for (int i=0; i<myvector.size(); i++) {
    cout << "i: " << i << " myvector[i]: " << myvector[i] << endl;
  }
  exit (0);
  */


}
```

Here is the caller graph for this function:



4.36.3.2 void fullTest ( )

Tests that require a full sandbox object including MTHREAD. Normally empty.

Definition at line 551 of file Sandbox.cpp.

Referenced by printAString(), and Init::setInitLevel1().

```
                  {

/*
  // Getting forest area by each forest type
  vector<int> regIds2 = MTHREAD->MD->getRegionIds(2);
  for(uint r=0;r<regIds2.size();r++){
    int rId = regIds2[r];
    ModelRegion* reg = MTHREAD->MD->getRegion(regIds2[r]);
    vector <string> fTypes= MTHREAD->MD->getForTypeIds();
    for(uint f=0;f<fTypes.size();f++){
      string ft = fTypes[f];
      forType* FT = MTHREAD->MD->getForType(ft);
      double totalArea = 0.0;
      vector <Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(regIds2[r]);
      for (uint p=0;p<rpx.size();p++){
        Pixel* px = rpx[p];
        totalArea +=  px->getDoubleValue (FT->forLayer, true);
      }
      cout << rId << "\t" << ft << "\t" << totalArea << endl;
    }
  }
  exit(1);
*/

  /*
  // Testing the new getForTypeParents()function
  vector<string>  parents = MTHREAD->MD->getForTypeParents();
  for(uint i=0;i<parents.size();i++){
    vector <string> childIds = MTHREAD->MD->getForTypeChilds(parents[i]);
    vector <int> childPos = MTHREAD->MD->getForTypeChilds_pos(parents[i]);
    double debug = 0.0;
  }
  */

  /*
  // Testing the reg->getArea() functions
  // Actually this need to be run further later, as pixels doesn't yet have area information
```

```
vector <string> dClasses = MTHREAD->MD->getStringVectorSetting("dClasses");
vector <string> fTypes= MTHREAD->MD->getForTypeIds();
ModelRegion* REG = MTHREAD->MD->getRegion(11041);
cout << "Total ft area: "<< REG->getArea()<< endl;

for(uint j=0;j<fTypes.size();j++){
  cout << fTypes[j] << "\t" << REG->getArea(fTypes[j]) << "\t" << REG->getArea(j) << endl;
}
for(uint j=0;j<fTypes.size();j++){
  cout << fTypes[j] << "\t" << REG->getArea(fTypes[j]) << "\t";
  for(uint u=0;u<dClasses.size();u++){
    cout << REG->getArea(j,u) << " ";
  }
  cout << endl;
}
*/

/*
// Testing getForData() function with no forest id specified
double vartest= MTHREAD->MD->getForData("forestChangeAreaIncrementsRel",11061,"","",2009);
cout << vartest << endl;
exit(0);
*/


/*
// Testing the decay model  - ok, passed
double initialValue = 100;
double halfLife = 2;
double years = 0;
double remStock = MTHREAD->CBAL->getRemainingStock(initialValue, halfLife, years); ///< Apply a single exponential decay model to retrieve the remining stock given the initial stock, the half life and the time passed from stock formation.
cout << "Remaining stock: " << remStock << endl;
years = 1;
remStock = MTHREAD->CBAL->getRemainingStock(initialValue, halfLife, years);
cout << "Remaining stock: " << remStock << endl;
years = 5;
remStock = MTHREAD->CBAL->getRemainingStock(initialValue, halfLife, years);
cout << "Remaining stock: " << remStock << endl;
years =10;
remStock = MTHREAD->CBAL->getRemainingStock(initialValue, halfLife, years);
cout << "Remaining stock: " << remStock << endl;
years = 200;
remStock = MTHREAD->CBAL->getRemainingStock(initialValue, halfLife, years);
cout << "Remaining stock: " << remStock << endl;
*/

/*
// Testing normSample
// template <typename K> K normSample (const K& avg, const K& stdev, const K& minval=NULL, const K& maxval=NULL)
// template <typename K> K normSample (const normal_distribution<K>& d, const std::mt19937& gen, const K& minval=NULL, const K& maxval=NULL)
double avg = 0.8;
double stdev = 0.2;
double minval = 0.0;
double maxval = 1.0;
double result;

cout << "Starting first method.." << endl;
normal_distribution<double> d(avg,stdev);
std::mt19937 gen = *MTHREAD->gen;
for (uint i=0;i<1000;i++){
  result = normSample(d, gen, minval, maxval);
  cout << "Result1: " << result << endl;
}
cout << "Finished first method and starting second one.." << endl;
for (uint i=0;i<1000;i++){
  result = normSample(avg, stdev, minval, maxval);
  cout << "Result2: " << result << endl;
}
cout << "Finished second method."<< endl;

exit(0);
*/


 //double disttest = MTHREAD->MD->getProdData("dist",11042,"",DATA_NOW,i2s(11061));
 //cout << disttest << endl;
 //exit(0);


 /*double test = MTHREAD->CBAL->getStock(11061, STOCK_INV);
 //STOCK_INV  -> from inventary source and death trees
 //STOCK_EXTRA  -> from inventary source and death trees
 //STOCK_PRODUCTS -> from products
```

```
  cout << "DONE" << endl;
  exit(0);
  */

   /*
   // Testing if forestData can uses other arbitrary elements in the diameter field in order to generalise it
   double test = MTHREAD->MD->getForData("covar",11082,"con_highF","con_highF");
   MTHREAD->MD->setForData(0.1,"covar",11082,"con_highF","con_highF");
   MTHREAD->MD->setForData(0.1,"covar",11061,"con_highF","con_highF",DATA_NOW,true);
   test = MTHREAD->MD->getForData("covar",11082,"con_highF","con_highF");
   test = MTHREAD->MD->getForData("covar",11061,"con_highF","con_highF");
   test = MTHREAD->MD->getForData("covar",11082,"con_highF","");
   cout << test << endl;
   exit(0);
   */

   /*
   // Testing getProdData for the freeDimension
   MTHREAD->MD->setProdData(0.4,"rt",11041,"hardWSawnW",DATA_NOW,true,"11061");
   MTHREAD->MD->setProdData(0.3,"rt",11041,"hardWSawnW",DATA_NOW,true,"11030");
   MTHREAD->MD->setProdData(0.2,"rt",11041,"hardWSawnX",DATA_NOW,true,"11030");
   double debug = MTHREAD->MD->getProdData("rt",11041,"hardWSawnW",DATA_NOW,"11061");
   double debug2 = MTHREAD->MD->getProdData("rt",11041,"hardWSawnW",DATA_NOW);
   cout << debug << "    " << debug2 << endl;
   exit(0);
   */

   /*
   // Testing api to call generic forest type data, parent/child
   cout << "Hello world " << endl;
   cout << MTHREAD->MD->getForData("freq_norm",11041,"broadL","",2040) << endl;
   MTHREAD->MD->setForData(100,"freq_norm",11041,"broadL","",2040);
   cout << MTHREAD->MD->getForData("freq_norm",11041,"broadL","",2040) << endl;
   cout << MTHREAD->MD->getForTypeParentId("broadL_highF")<< endl;
   cout << MTHREAD->MD->getForTypeParentId("con_highF")<< endl;
   exit(0);
   */

   /*
   // Testing for each region how far is the average of the multipliers from 1
   vector<int>  regIds  =    MTHREAD->MD->getRegionIds(2);
   vector <string> ftypes = MTHREAD->MD->getForTypeIds();

   cout << "*** Checking how far is the tpMultiplier far from 1 in each region:" << endl;
   for (int i=0;i< regIds.size();i++){
       ModelRegion* region = MTHREAD->MD->getRegion(regIds[i]);
       vector <Pixel*>  regpixels =   MTHREAD->GIS->getAllPlotsByRegion(regIds[i]);
       if(regpixels.size()==0) continue;
       cout << "*** " << region->getRegLName() << ":  "<< endl;
       for(int ft = 0;ft<ftypes.size();ft++){
           double tot = 0;
           double avg = 0;
           for(int j=0;j<regpixels.size();j++){
             tot += regpixels[j]->getSpModifier(ftypes[ft]);
           }
           avg = tot/regpixels.size();
           cout << ftypes[ft] << ":  " << avg << endl;
       }
   }
   exit(0);
   */

   /*
   // Testing the number of plots in the model
   vector <ModelRegion*> regions = MTHREAD->MD->getAllRegions();
   int total = 0;
   cout << "*** Pixels by region:" << endl;
   for (int i=0;i< regions.size();i++){
       vector <Pixel*>  regpixels =   MTHREAD->GIS->getAllPlotsByRegion(*regions[i]);
       cout << regions[i]->getRegLName() << ":  " << regpixels.size() << endl;
       total += regpixels.size() ;
   }
   cout << "** Total:  " << total << endl;
   exit(0);
   */

 /*
 // Testing the new random distributions. Requires the pointer MTHREAD->gen to be initialised,
 // so this test can’t run in basic test.
 std::normal_distribution<double> d(100000,3); // default any how to double
 for(int n=0; n<20; ++n) {
   double x = d(*MTHREAD->gen);
   int i = round(d(*MTHREAD->gen));
   cout << i << ’;’ << 1 << endl;
 }
 exit (0);
```

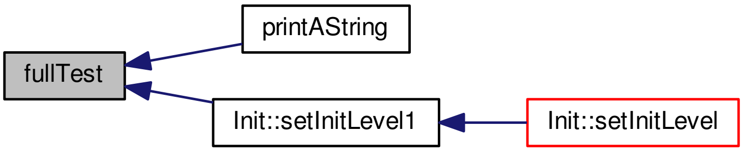
```
  */

  /*
  // Testing I have correctly the info about world price !!!
  // yes, it seems ok here !!!
  int firstYear = MTHREAD->MD->getIntSetting("initialYear");
  int initialOptYear= MTHREAD->MD->getIntSetting("initialOptYear");
  int simulationYears = MTHREAD->MD->getIntSetting("simulationYears");
  int WL2 = MTHREAD->MD->getIntSetting("worldCodeLev2");
  vector <string> priProducts = MTHREAD->MD->getStringVectorSetting("priProducts");
  vector <string> secProducts = MTHREAD->MD->getStringVectorSetting("secProducts");
  vector <string> allProducts = priProducts;
  allProducts.insert( allProducts.end(), secProducts.begin(), secProducts.end() );

  for(uint i=0;i<allProducts.size();i++){
    for(int y=firstYear; y<initialOptYear+simulationYears; y++){
      double pw = MTHREAD->MD->getProdData("pl",WL2,allProducts[i],y);
      cout << allProducts[i] << "  " << y << "  " << pw << endl;
    }
  }
  exit (0);
  */

  /*
  // testing Pixel::getMultiplier (const string& multiplierName, const string& forName, int year)
  Pixel* px = MTHREAD->GIS->getPixel(0);
  double debug1 = px->getMultiplier("tp_multiplier","broadL_highF",2012);
  double debug2 = px->getMultiplier("tp_multiplier","broadL_highF",2008);
  double debug3 = px->getMultiplier("tp_multiplier","broadL_highF",2009);
  double debug4 = px->getMultiplier("tp_multiplier","broadL_highF",2010);
  double debug5 = px->getMultiplier("mortCoeff_multiplier","broadL_highF",2012);
  double debug6 = px->getMultiplier("mortCoeff_multiplier","con_copp",2012);
  double debug7 = px->getMultiplier("blaaaa","broadL_highF",2012);

  double debug10 = debug1;
*/

  /*
  // testing reading a directory
  string dir = MTHREAD->MD->getBaseDirectory()+MTHREAD->MD->getStringSetting("spatialDataSubfolder");
  vector<string> files = vector<string>();

  MTHREAD->MD->getFilenamesByDir (dir,files, ".grd");

  for (unsigned int i = 0;i < files.size();i++) {
    cout << files[i] << endl;
  }
  */

  /*
  // testing ModelData:: ModelData::calculateAnnualisedEquivalent(double amount_h, int years_h)
  cout << "Value:" << MTHREAD->MD->calculateAnnualisedEquivalent(500.,4) << endl;
  cout << "Value:" << MTHREAD->MD->calculateAnnualisedEquivalent(500.,30) << endl;
  cout << "Value:" << MTHREAD->MD->calculateAnnualisedEquivalent(107.035040105,10) << endl;
  cout << "Value:" << MTHREAD->MD->calculateAnnualisedEquivalent(8.91507,1) << endl;
  cout << "Done" << endl;
  exit(0);
  */

  /*
  // testing snprintf
  vector <int> myintegers;
  vector <double> mydoubles;
  char  szResult[24];

  myintegers.push_back(1);
  myintegers.push_back(202);
  myintegers.push_back(3003);
  myintegers.push_back(400004);
  myintegers.push_back(50000005);
  myintegers.push_back(6000000006);
  mydoubles.push_back(1.1234567890);
  mydoubles.push_back(12345678.9);
  mydoubles.push_back(12345678.90123456);
  mydoubles.push_back(6000000006.123456789012);
  for(uint i=0;i<myintegers.size();i++){
    snprintf ( szResult, sizeof(szResult), "%d", myintegers[i] );  // "safe" version
    cout << "int/string: " << myintegers[i] << "  /  " << szResult << endl;
  }
  for(uint i=0;i<mydoubles.size();i++){
      snprintf ( szResult, sizeof(szResult), "%f", mydoubles[i] );  // "safe" version
    cout << "double/string: " << mydoubles[i] << "  /  " << szResult << endl;
  }
  exit(0);
  */

  /*
  // testing stod() ..
  // this is giving different results if gui or console mode !!
  vector<string> numbers;
  numbers.push_back("123.1234567890");
  numbers.push_back("123.1234");
  numbers.push_back("123,1234567890");
  numbers.push_back("123,1234");
  double outd;
  for(uint i=0;i<numbers.size();i++){
    try {
      outd =  stod(numbers[i]);
      cout << "Conversion passed: " << numbers[i] << "  -  " << outd << endl;
    } catch (...) {
      cout << "Conversion DID NOT passed: " << numbers[i] << "  -  " <<endl;
    }
  }
  exit(0);
  */

/*
// ------------------------------------
// Testing makeKeyProdData() and unpackKeyProdData()
  string parName = "za";
  int regId = 20000;
  string prod = "wood";
  string freeDim = "";
  string key = MTHREAD->MD->makeKeyProdData(parName,i2s(regId),prod,freeDim);
  cout << "key: " << key << endl;
  MTHREAD->MD->unpackKeyProdData(key,parName,regId,prod,freeDim);
  cout << "parName: " << parName << endl;
  cout << "regId: " << regId << endl;
  cout << "prod: " << prod << endl;
  cout << "freeDim: " << freeDim << endl;
  exit(0);
*/

/*
// ---------------------------------------------
// checking the functions dataMapCheckExist() and dataMapGetValue() works well
typedef map<string, vector <double> > DataMap;
typedef pair<string, vector <double> > DataPair;

vector <double> abaa (5, 1.);
vector <double> abcc (5,10);
vector <double> anbb (5,100);
vector <double> andd (5,5);
vector <double> anff (5,3);
vector <double> ag (5,2);
vector <double> agii (5,7);



DataMap dM;
dM.insert(DataPair("abaa", abaa));
dM.insert(DataPair("abcc", abcc));
dM.insert(DataPair("anbb", anbb));
dM.insert(DataPair("andd", andd));
dM.insert(DataPair("anff", anff));
dM.insert(DataPair("ag", ag));
dM.insert(DataPair("agii", agii));

vector<string> tests;
tests.push_back("ab");
tests.push_back("anbb");
tests.push_back("ane");
tests.push_back("an");
tests.push_back("ac");
tests.push_back("ag");
tests.push_back("agii");
tests.push_back("al");


bool found;
double value;

for(uint i=0;i<tests.size();i++){
  found = MTHREAD->MD->dataMapCheckExist(dM, tests[i]);
  value = MTHREAD->MD->dataMapGetValue(dM, tests[i],2010);
  cout << tests[i] << ": " << b2s(found) << " value: "<< value << endl;
}

exit(0);
*/


  /*
  // testing how to search on a vector using the find algorithm

  vector<string> names;
  names.push_back("pippo");
  names.push_back("topolino");
  names.push_back("minni");
  names.push_back("paperino");

  string toSearch1 = "minni";
  string toSearch2 = "zio paperone";

  if( find(names.begin(), names.end(), toSearch1)!= names.end() ){
    cout << "minni trovata" << endl;
  }
    if( find(names.begin(), names.end(), toSearch2)!= names.end() ){
    cout << "zio paperone trovato" << endl;
  }
  cout << "test on find ended." << endl;
  exit(0);
  */

// ------------------------------------------------------------------


  /*
  int a;
  a = getSetting<int>("myIntData", TYPE_INT);

  string b;
  b = getSetting<string>("myStringData", TYPE_STRING);

  bool c;
  c = getSetting<bool>("myBoolData", TYPE_BOOL);


  cout << "A is: " << a << endl;

  cout << "B is: " << b << endl;

  cout << "C is: " << c << endl;

  //vector<string> getVectorSetting <string> ("test", TYPE_STRING);
  //template <class T> vector <T> getVectorSetting(string name_h, int type);

  //vector <string> myStrings = getVectorSetting <vector<string> > ("test", TYPE_STRING);

  string s = GccTest("test");
  int i = GccTest("test");
  vector <int> iVector = GccTest("test");

  for (int i=0; i< iVector.size(); i++){
    cout << "iVector: " << iVector.at(i) << endl;
  }
  */

  // ------------------------------------------------------------------

  /* // I learned: how to access elements - both objects and pointers - of a vector using pointers
  // testing how to operate with iterators over a pointer element in an array:

  cout << "Starting iterator test..." << endl;

  TestStructure a,b,c,d;
  a.i=0; b.i=1; c.i=2; d.i=3;
  TestStructure* ap;
  TestStructure* bp;
  TestStructure* cp;
  TestStructure* dp;

  ap = &a;
  bp = &b;
  cp = &c;
  dp = &d;

  vector <TestStructure>  objects;
  vector <TestStructure*> pointers;

  objects.push_back(a);
  objects.push_back(b);
  objects.push_back(c);
  objects.push_back(d);

  pointers.push_back(ap);
  pointers.push_back(bp);
  pointers.push_back(cp);
  pointers.push_back(dp);

  vector<TestStructure>::iterator pi;
```

```
vector<TestStructure*>::iterator pp;

 //ok it works
//for ( pi = objects.begin() ; pi != objects.end();){
//  if(pi->i==2){
//    objects.erase(pi);
//  }
//  else {
//    ++pi;
//  }
//}

//for (int j=0;j<objects.size();j++){
//  cout << j << " object is: " << objects[j].i << endl;
//}


// works as well ;-))
for ( pp = pointers.begin() ; pp != pointers.end();){
  if( (*pp)->i==2){
    //delete (*pp);
    pointers.erase(pp);
  }
  else {
    ++pp;
  }
}

for (int j=0;j<pointers.size();j++){
  cout << j << " pointers is: " << pointers[j]->i << endl;
}

// c is not destructed if we don't explicitelly call delete over the pointer...
cout << c.i << endl; // this go in seg-frag if we call delete (*pp)..
*/

// ---------------------------------------------------------------------
/* test on how to remove from a map.. deletable
map <int, string> test;
test.insert(pair<int, string>(2, "pippo"));
test.insert(pair<int, string>(1, "pluto"));
test.insert(pair<int, string>(5, "minni"));
test.insert(pair<int, string>(3, "topolino"));


map <int, string>::iterator p;
p=test.find(3);
if(p != test.end()){
  cout << p->second <<endl;
  test.erase(p);
}
else {
  cout << "not found " << endl;
}

map <int, string>::iterator p2;
p2=test.find(3);
if(p2 != test.end()){
  cout << p2->second <<endl;
  test.erase(p2);
}
else {
  cout << "not found " << endl;
}
*/

 /*vector<int> test;
 for (int i=0;i<5;i++) test.push_back(i);
 cout << "test.." << endl;
for (uint i=0;i<test.size();i++){
   cout << "Test "<<i<<": "<<test.at(i) << endl;
}
//test.erase(2);

vector<int>::iterator p;
for ( p = test.begin() ; p != test.end();){
  if(*p == 1 || *p == 2 || *p==4){
    test.erase(p);
  }
  else {
    ++p;
  }
}


for (uint i=0;i<test.size();i++){
   cout << "Test "<<i<<": "<<test.at(i) << endl;
  }

//  test.erase(remove_if(test.begin(), test.end(), FindMatchingString(&fs))

//  for (int i=0;i<test.size();i++) cout << "TEST: "<<i<< " " << test.at(i) << endl;
  */

  /*
  // On this test I am showing how to "move" one pointer from a vector of pointers to an other one. The real case is used to move Agent_farmer* pointers from the managedAgents vector to the removedVector.

  double* myDouble1 = new double(1);
  double* myDouble2 = new double(2);
  double* myDouble3 = new double(3);

  vector <double*> origin;
  vector <double*> destination;

  origin.push_back(myDouble1);
  origin.push_back(myDouble2);
  origin.push_back(myDouble3);

  cout << "MyDouble2: "<< *myDouble2 << endl;
  vector<double*>::iterator doublePIterator;

  for (int i=0;i<origin.size();i++){
    cout << i << " origin is: " << *origin[i] << endl;
  }

  for ( doublePIterator = origin.begin() ; doublePIterator !=origin.end();){
    if(*doublePIterator == myDouble2){
      destination.push_back(myDouble2);
      origin.erase(doublePIterator);
    }
    else {
      ++doublePIterator;
    }
  }

  for (int i=0;i<origin.size();i++){
    cout << i << " origin is now: " << *origin[i] << endl;
  }

  for (int i=0;i<destination.size();i++){
    cout << i << " destination is: " << *destination[i] << endl;
  } */

  // -----------------------------------------------------------------
  /*
  // Test on how to return a vector of pointers from a member vector of data
  TestStructure a,b,c,d;
  a.i=0; b.i=1; c.i=2; d.i=3;
  testVector.push_back(a);
  testVector.push_back(b);
  testVector.push_back(c);
  testVector.push_back(d);

  vector<TestStructure*>  myVector=getTestStructure();

  for(uint i=0;i<myVector.size();i++){
    msgOut(MSG_DEBUG, i2s(myVector[i]->i));
  }
  */

  /*
  // Deleting an object and inserting a new one on a vector of objects.. it doesn't works.. problems with the last element..
  vector<BasicData>::iterator p;
  for(p=programSettingsVector.begin();p!=programSettingsVector.end();p++){
    if(p->name == SETT.name){
      programSettingsVector.erase(p);
      programSettingsVector.insert(p,1,SETT);
      cout << SETT.name <<endl;
      break;
    }
  }
  */

  /*double test = -987654321.987654321;
  double result;
  result = fabs(test);
  cout << "Test: "<< result << endl;*/


  /*
  // Testing the zip library:

```

```
cout <<"Hello world Zip!" << endl;

QString file = "data/testInput.ods";
QString out = "data/tempInput";
QString pwd = "";
if (!QFile::exists(file))
{
  cout << "File does not exist." << endl << endl;
  //return false;
}

UnZip::ErrorCode ec;
UnZip uz;

if (!pwd.isEmpty())
  uz.setPassword(pwd);

ec = uz.openArchive(file);
if (ec != UnZip::Ok)
{
      //cout << "Failed to open archive: " << uz.formatError(ec).toAscii().data() << endl << endl; // Qt4
      cout << "Failed to open archive: " << uz.formatError(ec).toLatin1().data() << endl << endl;  // Qt5
  //return false;
}

ec = uz.extractAll(out);
if (ec != UnZip::Ok)
{
      //cout << "Extraction failed: " << uz.formatError(ec).toAscii().data() << endl << endl; // Qt4
      cout << "Extraction failed: " << uz.formatError(ec).toLatin1().data() << endl << endl;  // Qt5
  uz.closeArchive();
  //return false;
}
*/

/*
// How to : delete an element from an array from its position
cout << "How to : delete an element from an array from its position" << endl;

vector <string> headers;
vector < vector <string> > records;
vector <string> firstrecord;
vector <string> secondrecord;
records.push_back(firstrecord);
records.push_back(secondrecord);

headers.push_back("a");
headers.push_back("b");
headers.push_back("");
headers.push_back("d");
headers.push_back("e");
headers.push_back("");

records[0].push_back("0");
records[0].push_back("1");
records[0].push_back("2");
records[0].push_back("3");
records[0].push_back("4");
records[0].push_back("5");
records[1].push_back("00");
records[1].push_back("11");
records[1].push_back("22");
records[1].push_back("33");
records[1].push_back("44");
records[1].push_back("55");

for (int i=headers.size()-1;i>=0;i--){
  if(headers[i] == ""){
    headers.erase(headers.begin()+i);
    for (int j=0;j<records.size();j++){
      records[j].erase(records[j].begin()+i);
    }
  }
}
for(uint i=0;i<headers.size();i++){
  cout << headers.at(i) << " - " << records[0].at(i) << " - " << records[1].at(i) << endl;
}
cout << "done!" << endl;
*/

//testThreads();
/*vector<double> numbers;
double cumNumbers = 0.00;
numbers.push_back(0.40);
numbers.push_back(0.10);
numbers.push_back(0.20);
numbers.push_back(0.08);
```

```
  numbers.push_back(0.22);

  for (uint i=0;i<numbers.size();i++){
    cumNumbers += numbers[i];
  }

  if (cumNumbers <= 0.99999999 || cumNumbers >= 1.00000001) {
    cout <<"Bastardo!"<<endl;
  } else {
    cout <<"qui funzia!"<<endl;
  }*/

}
```

Here is the caller graph for this function:



4.36.3.3 T getSetting (string *name_h,* int *type*)

Definition at line 1313 of file Sandbox.cpp.

```
                                          {

  string myIntData;
  myIntData = "34";
  string myStringData;
  myStringData = "abcdefg";

  string myBoolData;
  myBoolData = "false";

  if(type==TYPE_INT){
    istringstream iss(myIntData);
    T x;
    iss >> x;
    return x;
  }

  if(type==TYPE_STRING){
    istringstream iss(myStringData);
    T x;
    iss >> x;
    return x;
  }

  if(type==TYPE_BOOL){
    string tempBoolString;
    if (myBoolData == "1" || myBoolData == "true" || myBoolData == "True" || myBoolData == "TRUE" ||
myBoolData == "vero" || myBoolData == "Vero"|| myBoolData == "VERO"){
      tempBoolString = "1";
    }
    else if (myBoolData == "0" || myBoolData == "false" || myBoolData == "False" || myBoolData == "FALSE"
|| myBoolData == "falso" || myBoolData == "falso"|| myBoolData == "FALSO"){
      tempBoolString = "0";
    }
    else {
      msgOut(MSG_CRITICAL_ERROR, "Impossible conversion of "+myBoolData+" to bool!.
 Aborted.");
    }
    istringstream iss(tempBoolString);
    T x;
01350     iss >> x;
01351     return x;
01352   }
01353
01354
01355 }
```

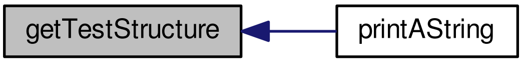
4.36.3.4 vector< **TestStructure** * > getTestStructure ( )

Definition at line 1367 of file Sandbox.cpp.

Referenced by printAString().

```
01367                         {
01368   vector <TestStructure*> toReturn;
01369   for (uint i=0;i<testVector.size();i++){
01370     //TestStructure* tempTest = new TestStructure;
01371     toReturn.push_back(&testVector[i]);
01372   }
01373   return toReturn;
01374
01375 }
```

Here is the caller graph for this function:



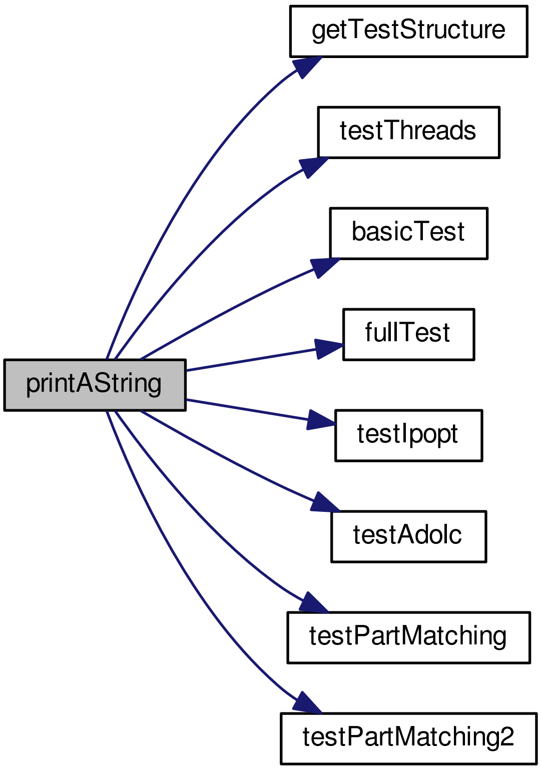
4.36.3.5 vector<T> getVectorSetting (string *name_h*, int *type*)

4.36.3.6 void printAString (string *what*) `[inline]`

Definition at line 50 of file Sandbox.h.

```
00050 {cout << "You printed: "<< what << endl;};
```

Here is the call graph for this function:



4.36.3.7 T test2 (const std::string & *s*)

Definition at line 1358 of file Sandbox.cpp.

```
                                  {
  std::istringstream iss(s);
  T x;
  iss >> x;
  return x;
}
```

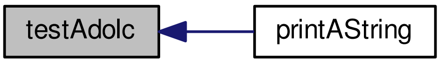
4.36.3.8 int testAdolc ( )

Definition at line 1553 of file Sandbox.cpp.

Referenced by printAString().

```
                 {

  using namespace Ipopt;
 // Create an instance of your nlp...
   SmartPtr<TNLP> myadolc_nlp = new MyADOLC_NLP();
  //SmartPtr<TNLP> myadolc_nlp = new MyADOLC_sparseNLP();

  // Create an instance of the IpoptApplication
  SmartPtr<IpoptApplication> app = new IpoptApplication();

  // Initialize the IpoptApplication and process the options
  ApplicationReturnStatus status;
  status = app->Initialize();
  if (status != Solve_Succeeded) {
    printf("\n\n*** Error during initialization!\n");
    return (int) status;
  }

  status = app->OptimizeTNLP(myadolc_nlp);

  if (status == Solve_Succeeded) {
    // Retrieve some statistics about the solve
    Index iter_count = app->Statistics()->IterationCount();
    printf("\n\n*** The problem solved in %d iterations!\n", iter_count);

    Number final_obj = app->Statistics()->FinalObjective();
    printf("\n\n*** The final value of the objective function is %e.\n", final_obj);
  }

  return (int) status;
}
```

Here is the caller graph for this function:



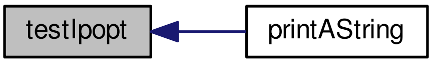
4.36.3.9 void testIpopt ( )

Definition at line 1502 of file Sandbox.cpp.

Referenced by printAString().

```
                  {


    using namespace Ipopt;

      // Create a new instance of your nlp
      //  (use a SmartPtr, not raw)
      SmartPtr<TNLP> mynlp = new Ipopt_nlp_problem_debugtest();

      // Create a new instance of IpoptApplication
      //  (use a SmartPtr, not raw)
      // We are using the factory, since this allows us to compile this
      // example with an Ipopt Windows DLL
      SmartPtr<IpoptApplication> app = IpoptApplicationFactory();

      // Change some options
      // Note: The following choices are only examples, they might not be
      //       suitable for your optimization problem.
      app->Options()->SetNumericValue("tol", 1e-7);
      app->Options()->SetStringValue("mu_strategy", "adaptive");
      app->Options()->SetStringValue("output_file", "ipopt.out");
      //app->Options()->SetStringValue("hessian_approximation", "limited-memory");
      //app->Options()->SetStringValue("derivative_test", "second-order");
      //app->Options()->SetStringValue("derivative_test_print_all", "yes");


      // The following overwrites the default name (ipopt.opt) of the
      // options file
      // app->Options()->SetStringValue("option_file_name", "hs071.opt");

      // Intialize the IpoptApplication and process the options
      ApplicationReturnStatus status;
      status = app->Initialize();
      if (status != Solve_Succeeded) {
        std::cout << std::endl << std::endl << "*** Error during initialization!" << std::endl;
        //return (int) status; // here the abort
      }

      // Ask Ipopt to solve the problem
      status = app->OptimizeTNLP(mynlp);

      if (status == Solve_Succeeded) {
        std::cout << std::endl << std::endl << "*** The problem solved!" << std::endl;
      }
      else {
        std::cout << std::endl << std::endl << "*** The problem FAILED!" << std::endl;
      }

}
```

Here is the caller graph for this function:



4.36.3.10 void testPartMatching ( )

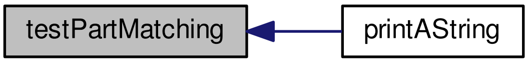
How to partial matching the key of a map.

Definition at line 1628 of file Sandbox.cpp.

Referenced by printAString().

```
01628                          {
01629 
01630     TStrStrMap tMap;
01631 
01632     tMap.insert(TStrStrPair("John", "AA"));
01633     tMap.insert(TStrStrPair("Mary", "BBB"));
01634     tMap.insert(TStrStrPair("Mother", "A"));
01635     tMap.insert(TStrStrPair("Moliere", "D"));
01636     tMap.insert(TStrStrPair("Marlon", "C"));
01637 
01638     testSearchMap(tMap, "Marl");
01639     testSearchMap(tMap, "Mo");
01640     testSearchMap(tMap, "ther");
01641     testSearchMap(tMap, "Mad");
01642     testSearchMap(tMap, "Mom");
01643     testSearchMap(tMap, "Perr");
01644     testSearchMap(tMap, "Jo");
01645 
01646   exit(0);
01647     return;
01648 }
```

Here is the caller graph for this function:



4.36.3.11 void testPartMatching2 ( )

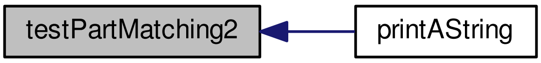
How to partial matching the key of a map.

Definition at line 1665 of file Sandbox.cpp.

Referenced by printAString().

```
01665                          {
01666 
01667     TStrStrMap tMap;
01668 
01669 
01670   tMap.insert(TStrStrPair("mortCoeff_multiplier#broadL_highF##2005", "2005"));
01671   tMap.insert(TStrStrPair("regLev_1", "-9999"));
01672   tMap.insert(TStrStrPair("regLev_2", "-9999"));
01673   tMap.insert(TStrStrPair("tp_multiplier#broadL_copp##2005", "-9999"));
01674   tMap.insert(TStrStrPair("tp_multiplier#broadL_highF##2005", "50"));
01675   tMap.insert(TStrStrPair("tp_multiplier#broadL_highF##2010", "2010"));
01676   tMap.insert(TStrStrPair("tp_multiplier#broadL_mixedF##2005", "-9999"));
01677   tMap.insert(TStrStrPair("tp_multiplier#con_copp##2005", "-9999"));
01678   tMap.insert(TStrStrPair("tp_multiplier#con_highF##2005", "-9999"));
01679   tMap.insert(TStrStrPair("tp_multiplier#con_mixedF##2005", "aa"));
01680 
01681   TStrStrMap::const_iterator i;
01682 
01683   for(i=tMap.begin();i!=tMap.end();i++){
01684     cout << i->first << ", " << i->second << endl;
01685   }
01686   cout << endl;
01687 
01688     testSearchMap2(tMap, "mortCoeff_multiplier#broadL_highF##2006");
01689     testSearchMap2(tMap, "tp_multiplier#broadL_highF##2008");
01690     testSearchMap2(tMap, "aaaaaa");
01691     testSearchMap2(tMap, "zzzzzz");
01692 
01693   exit(0);
01694     return;
01695 }
```

Here is the caller graph for this function:



4.36.3.12 void testSearchMap (const map< string, string > & *map*, const string & *search_for*) [private]

Definition at line 1613 of file Sandbox.cpp.

```
01613                                                                        {
01614   TStrStrMap::const_iterator i = map.lower_bound(search_for);
01615   for(;i != map.end();i++){
01616     const string& key = i->first;
01617     if (key.compare(0, search_for.size(), search_for) == 0) {// Really a prefix?
01618         cout << i->first << ", " << i->second << endl;
01619     } else {
01620       break;
01621     }
01622   }
01623 
01624 }
```

4.36.3.13 void testSearchMap2 (const map< string, string > & *map_h*, const string & *search_for*) [private]

Definition at line 1651 of file Sandbox.cpp.

```
01651                                                                           {
01652   TStrStrMap::const_iterator i = map_h.upper_bound(search_for);
01653   if(i!= map_h.begin())  i--;
01654   const string& key = i->first;
01655   string search_base = search_for.substr(0,search_for.size()-4);
01656   if (key.compare(0, search_base.size(), search_base) == 0){
01657     cout << "MATCH: " << search_for <<", "<< i->first << ", " << i->second << endl;
01658   } else {
01659     cout << "NOTM:  " << search_for <<", "<< i->first << endl;
01660   }
01661 
01662 }
```

4.36.3.14 void testThreads ( )

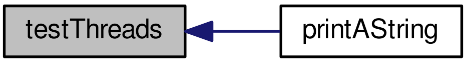
Definition at line 1380 of file Sandbox.cpp.

Referenced by printAString().

```
01380                      {
01381 
01382         /*
01383         PSEUDOCODE
01384         - attivo i vari thread
01385         - per ogni closestAgent itero fra i vari thread e se "è libero" gli assegno il closestAgent
01386         - quando ho finito i closestAgent aspetto che tutti i threads abbiano finito il lavoro
01387         - chiudo i threads
01388         - vado avanti
01389         */
01390         int nAgents= 50;
01391         vector<TestStructure*> myAgents;
01392         vector<double> myResults (nAgents, (double) 0);
01393         //int nThreads = MTHREAD->MD->getIntSetting("nThreads");
01394         int nThreads= 5;
01395 
01396         for (int i=0; i < nAgents; i++){
01397           TestStructure* myAgent = new TestStructure;
01398           myAgent->i = i;
01399           myAgent->random =  (0+( (double)rand() / ((double)(RAND_MAX)+(double)(1)) )*(10-0+1))/ (
      double)100;
01400           myAgents.push_back(myAgent);
01401         }
01402 
01403         vector <testThread*> myThreads ;
01404 
01405         for (int i=0; i < nThreads; i++){
01406           testThread* myThread = new testThread;
01407           myThreads.push_back(myThread);
01408         }
01409 
01410         for (uint i=0;i<myAgents.size();i++){
```

```
          bool assigned = false;
          while(!assigned) {
            for (uint j=0;j<myThreads.size();j++){
              if (!myThreads[j]->isRunning()){
                cout << "Assigning agent " << i << " to thread " << j << endl;
                myThreads[j]->assignJob(myAgents[i]);
                myThreads[j]->start();
                assigned = true;
                break;
              }
              else {
                cout << "Thread " << j << " is busy" << endl;
              }
            }
          }
        }
        /*
        volatile bool somethingStopping = true;
        while (somethingStopping){
          somethingStopping = false;
          for (uint i=0;i<myThreads.size();i++){
            if(myThreads[i]->isRunning()){
              somethingStopping = true;
              //cout << "somethingStopping is true" << endl;
            }
          }
        }

        if (somethingStopping) {
          cout << "somethingStopping is true" << endl;
        }
        else {
          cout << "somethingStopping is false" << endl;
        }
        cout << "pinco pallo sono nel mezzo dei threads..."<<endl;
        */
        for (int i=0; i < nThreads; i++){
          myThreads[i]->wait();
        }


        for (int i=0; i < nThreads; i++){
          delete myThreads[i];
        }

        for (uint i=0;i<myAgents.size();i++){
          //cout <<myAgents[i]->cachedOffer<<endl;

          double random =  (0+( (double)rand() / ((double)(RAND_MAX)+(double)(1)) )*(10-0+1))/ (double)100;

          // important !
          // for random integer see also std::uniform_int_distribution :
          // http://stackoverflow.com/questions/7780918/stduniform-int-distributionint-range-in-g-and-msvc
          // in regmas:
          // int randomRed = int (50+( (double)rand() / ((double)(RAND_MAX)+(double)(1)) )*(255-50+1)); // randomRed is [50,255] Don't use "randomNumber % range" !!

          //cout <<random<<endl;
        }

        //thread1.stop();
        cout << "FINITO"<<endl;


}
```

Here is the caller graph for this function:



4.36.4 Member Data Documentation

4.36.4.1 vector< **TestStructure** > testVector `[private]`

Definition at line 61 of file Sandbox.h.

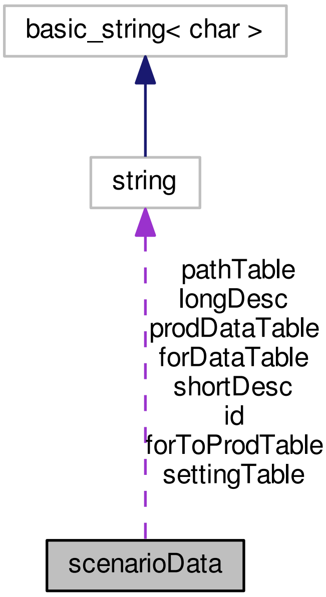
The documentation for this class was generated from the following files:

- /home/lobianco/git/ffsm_pp/src/Sandbox.h
- /home/lobianco/git/ffsm_pp/src/Sandbox.cpp

4.37 scenarioData Struct Reference

`#include <ModelData.h>`

Collaboration diagram for scenarioData:



Public Attributes

- string id
- string shortDesc
- string longDesc
- string settingTable
- string forDataTable
- string prodDataTable
- string forToProdTable
- string pathTable

4.37.1 Detailed Description

Definition at line 60 of file ModelData.h.

4.37.2 Member Data Documentation

4.37.2.1 string forDataTable

Definition at line 65 of file ModelData.h.

Referenced by ModelData::setScenarioData(), and ModelData::setScenarioForData().

4.37.2.2 string forToProdTable

Definition at line 67 of file ModelData.h.

Referenced by ModelData::setScenarioData(), and ModelData::setScenarioProductResourceMatrixLink().

4.37.2.3 string id

Definition at line 61 of file ModelData.h.

Referenced by ModelData::setScenarioData().

4.37.2.4 string longDesc

Definition at line 63 of file ModelData.h.

Referenced by ModelData::setScenarioData().

4.37.2.5 string pathTable

Definition at line 68 of file ModelData.h.

Referenced by ModelData::setScenarioData(), and ModelData::setScenarioPathogenRules().

4.37.2.6 string prodDataTable

Definition at line 66 of file ModelData.h.

Referenced by ModelData::setScenarioData(), and ModelData::setScenarioProdData().

4.37.2.7 string settingTable

Definition at line 64 of file ModelData.h.

Referenced by ModelData::setScenarioData(), and ModelData::setScenarioSettings().

4.37.2.8 string shortDesc

Definition at line 62 of file ModelData.h.

Referenced by ModelData::setScenarioData().

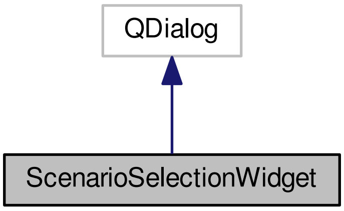
The documentation for this struct was generated from the following file:

- /home/lobianco/git/ffsm_pp/src/ModelData.h

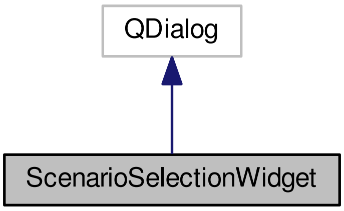
4.38 ScenarioSelectionWidget Class Reference

`#include <ScenarioSelectionWidget.h>`

Inheritance diagram for ScenarioSelectionWidget:



Collaboration diagram for ScenarioSelectionWidget:



Public Member Functions

- ScenarioSelectionWidget (QWidget ∗parent=0)
- void receiveScenarioOptions (const QVector< QString > &scenarios_h)

Public Attributes

- QComboBox ∗ scenarioSelector

Private Member Functions

- ∼ScenarioSelectionWidget ()

Private Attributes

- QLabel ∗ label

4.38.1 Detailed Description

Simple widget to show the available scenarios so that the user can choose one.

Author

Antonello Lobianco `antonello@regmas.org`

Definition at line 37 of file ScenarioSelectionWidget.h.

4.38.2 Constructor & Destructor Documentation

4.38.2.1 ScenarioSelectionWidget (QWidget ∗ *parent* = 0)

Definition at line 29 of file ScenarioSelectionWidget.cpp.

```
00029                                                                  : QDialog(parent) {
00030 
00031   label = new QLabel(tr("Select the scenario you want to run..."));
00032   scenarioSelector = new QComboBox();
00033   QVBoxLayout *mainLayout = new QVBoxLayout;
00034   mainLayout->addWidget(label);
00035   mainLayout->addWidget(scenarioSelector);
00036   setLayout(mainLayout);
00037   setWindowTitle(tr("Scenario selection"));
00038   setFixedHeight(sizeHint().height());
00039 
00040   //connect(scenarioSelector, SIGNAL( activated(const QString&)), this, SLOT( processSelectedScenario(const
      QString &)   ));
00041   //connect(scenarioSelector, SIGNAL( activated(const QString&)), this, SLOT( close()));
00042 
00043 }
```

4.38.2.2 ∼ScenarioSelectionWidget ( ) [private]

Definition at line 45 of file ScenarioSelectionWidget.cpp.

```
00045                                                {
00046 }
```

4.38.3 Member Function Documentation

4.38.3.1 void receiveScenarioOptions (const QVector< QString > & *scenarios_h*)

Definition at line 50 of file ScenarioSelectionWidget.cpp.

```
00050                                                                                  {
00051   scenarioSelector->clear();
00052   for (uint i=0; i< scenarios_h.size();i++){
00053     scenarioSelector->addItem(scenarios_h.at(i));
00054   }
00055   //scenarioSelector->setFocus(); // may be not visible, no effect!
00056   //scenarioSelector->grabMouse();
00057   //scenarioSelector->grabKeyboard();
00058 }
```

4.38.4 Member Data Documentation

4.38.4.1 QLabel∗ label `[private]`

Definition at line 46 of file ScenarioSelectionWidget.h.

Referenced by ScenarioSelectionWidget().

4.38.4.2 QComboBox∗ scenarioSelector

Definition at line 43 of file ScenarioSelectionWidget.h.

Referenced by receiveScenarioOptions(), and ScenarioSelectionWidget().

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

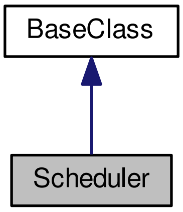
- /home/lobianco/git/ffsm_pp/src/ScenarioSelectionWidget.h
- /home/lobianco/git/ffsm_pp/src/ScenarioSelectionWidget.cpp

4.39 Scheduler Class Reference

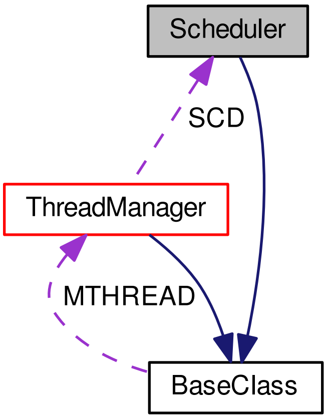
Manage the yearly loops.

`#include <Scheduler.h>`

Inheritance diagram for Scheduler:



Collaboration diagram for Scheduler:



Public Member Functions

- Scheduler (ThreadManager ∗MTHREAD_h)
- ∼Scheduler ()
- void run ()
- int getIteration ()
- int getYear ()
- int setYear (const int &year_h)
- int advanceYear ()

Private Attributes

- int iteration
- int year

Additional Inherited Members

4.39.1 Detailed Description

Manage the yearly loops.

This class is responsable to manage the time-dimension of the program.
It start its job when Init has ended and schedule the various operation to be done during the year loops.

Author

Antonello Lobianco

Definition at line 42 of file Scheduler.h.

4.39.2 Constructor & Destructor Documentation

4.39.2.1 Scheduler (ThreadManager ∗ *MTHREAD_h*)

Definition at line 32 of file Scheduler.cpp.

```
00032                                          {
00033   MTHREAD=MTHREAD_h;
00034   iteration=0;
00035 }
```

4.39.2.2 ∼Scheduler ( )

Definition at line 37 of file Scheduler.cpp.

```
00037                       {
00038 }
```

4.39.3 Member Function Documentation

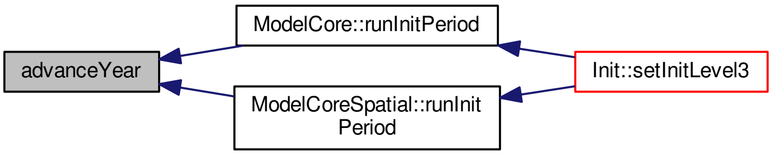
4.39.3.1 int advanceYear ( ) [inline]

Definition at line 51 of file Scheduler.h.

Referenced by ModelCore::runInitPeriod(), and ModelCoreSpatial::runInitPeriod().

```
00051 {year += 1;}
```

Here is the caller graph for this function:



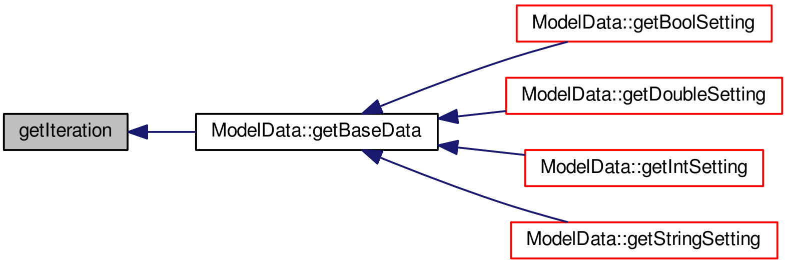
4.39.3.2 int getIteration ( ) [inline]

Definition at line 48 of file Scheduler.h.

Referenced by ModelData::getBaseData().

```
00048 {return iteration;};
```

Here is the caller graph for this function:


getIteration
ModelData::getBaseData
ModelData::getBoolSetting
ModelData::getDoubleSetting
ModelData::getIntSetting
ModelData::getStringSetting

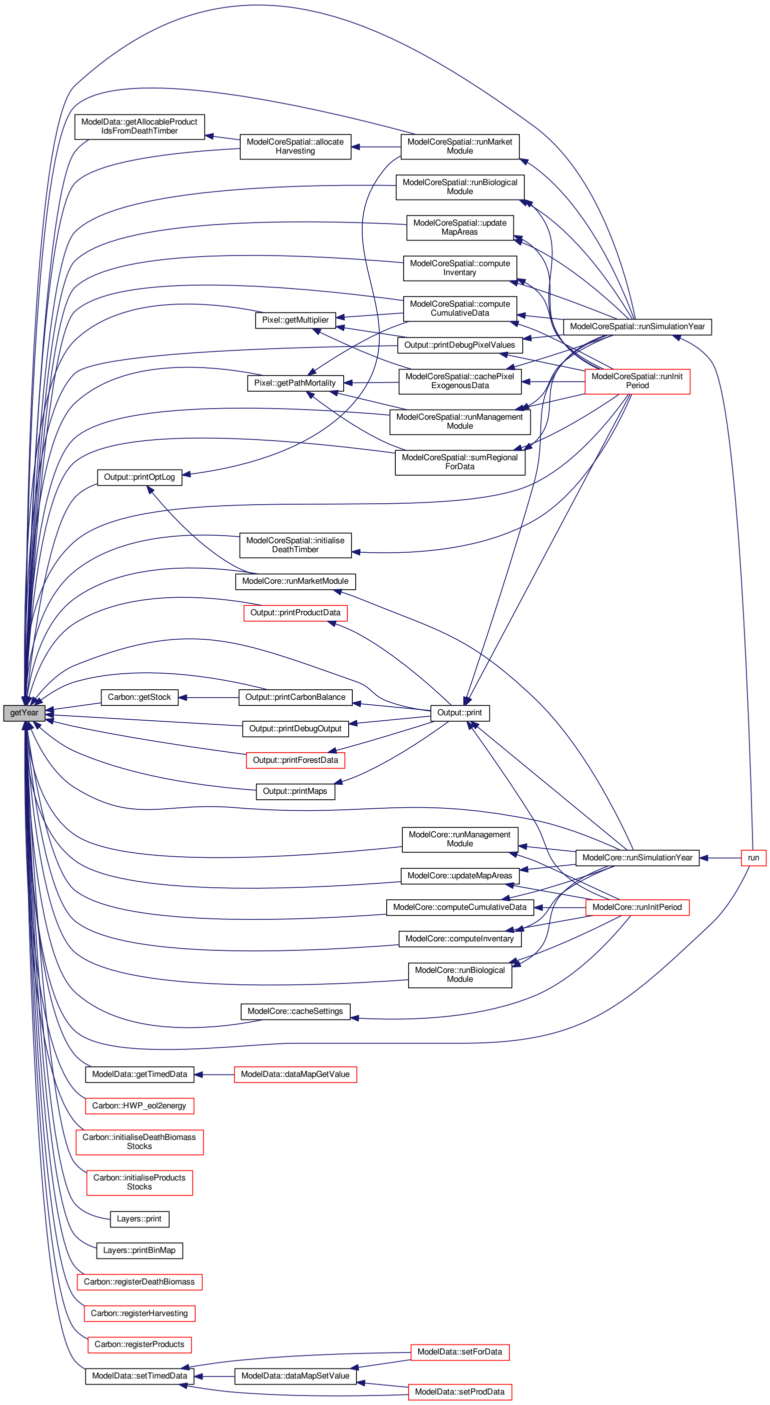
4.39.3.3 int getYear ( ) [inline]

Definition at line 49 of file Scheduler.h.

Referenced by ModelCoreSpatial::allocateHarvesting(), ModelCore::cacheSettings(), ModelCore::computeCumulativeData(), ModelCoreSpatial::computeCumulativeData(), ModelCore::computeInventary(), ModelCoreSpatial::computeInventary(), ModelData::getAllocableProductIdsFromDeathTimber(), Pixel::getMultiplier(), Pixel::getPathMortality(), Carbon::getStock(), ModelData::getTimedData(), Carbon::HWP_eol2energy(), Carbon::initialiseDeathBiomassStocks(), ModelCoreSpatial::initialiseDeathTimber(), Carbon::initialiseProductsStocks(), Output::print(), Layers::print(), Layers::printBinMap(), Output::printCarbonBalance(), Output::printDebugOutput(), Output::printDebugPixelValues(), Output::printForestData(), Output::printMaps(), Output::printOptLog(), Output::printProductData(), Carbon::registerDeathBiomass(), Carbon::registerHarvesting(), Carbon::registerProducts(), run(), ModelCore::runBiologicalModule(), ModelCoreSpatial::runBiologicalModule(), ModelCoreSpatial::runInitPeriod(), ModelCore::runManagementModule(), ModelCoreSpatial::runManagementModule(), ModelCore::runMarketModule(), ModelCoreSpatial::runMarketModule(), ModelCore::runSimulationYear(), ModelCoreSpatial::runSimulationYear(), ModelData::setTimedData(), ModelCoreSpatial::sumRegionalForData(), ModelCore::updateMapAreas(), and ModelCoreSpatial::updateMapAreas().

```
00049 {return year;}
```

Here is the caller graph for this function:


ModelData::getAllocableProductIdsFromDeathTimber
ModelCoreSpatial::allocateHarvesting
ModelCoreSpatial::runMarketModule
ModelCoreSpatial::runBiologicalModule
ModelCoreSpatial::updateMapAreas
ModelCoreSpatial::computeInventary
ModelCoreSpatial::computeCumulativeData
Pixel::getMultiplier
ModelCoreSpatial::runSimulationYear
Output::printDebugPixelValues
Pixel::getPathMortality
ModelCoreSpatial::cachePixelExogenousData
ModelCoreSpatial::runInitPeriod
ModelCoreSpatial::runManagementModule
ModelCoreSpatial::sumRegionalForData
Output::printOptLog
ModelCoreSpatial::initialiseDeathTimber
ModelCore::runMarketModule
Output::printProductData
Carbon::getStock
Output::printCarbonBalance
getYear
Output::print
Output::printDebugOutput
Output::printForestData
Output::printMaps
ModelCore::runManagementModule
ModelCore::runSimulationYear
run
ModelCore::updateMapAreas
ModelCore::computeCumulativeData
ModelCore::runInitPeriod
ModelCore::computeInventary
ModelCore::runBiologicalModule
ModelCore::cacheSettings
ModelData::getTimedData
ModelData::dataMapGetValue
Carbon::HWP_eol2energy
Carbon::initialiseDeathBiomassStocks
Carbon::initialiseProductsStocks
Layers::print
Layers::printBinMap
Carbon::registerDeathBiomass
Carbon::registerHarvesting
Carbon::registerProducts
ModelData::setForData
ModelData::setTimedData
ModelData::dataMapSetValue
ModelData::setProdData

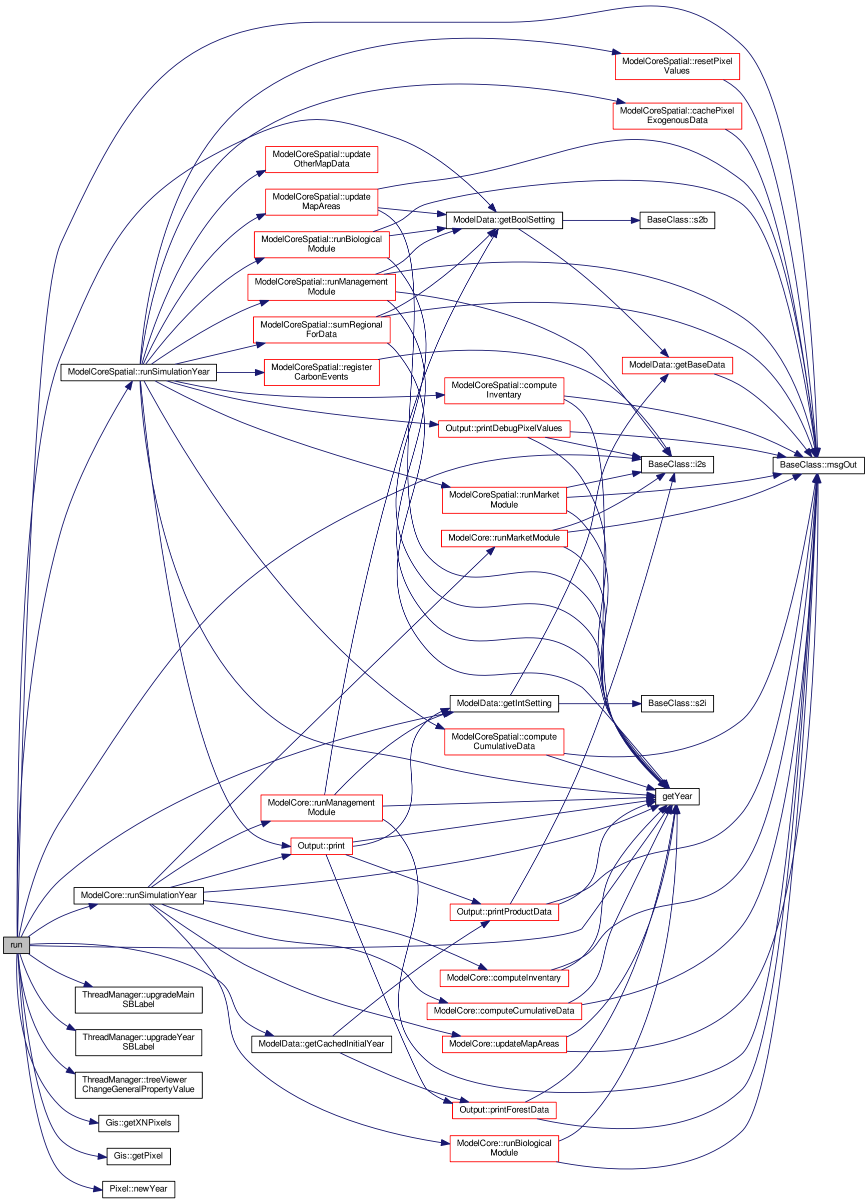
4.39.3.4 void run ( )

Definition at line 41 of file Scheduler.cpp.

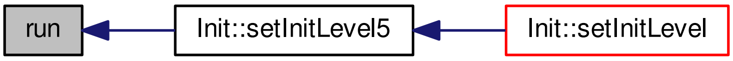
Referenced by Init::setInitLevel5().

```
              {

  int initialYear           = MTHREAD->MD->getIntSetting("initialYear");
  int initialSimulationYear = MTHREAD->MD->getIntSetting("initialOptYear");
  int preSimulationYears = initialSimulationYear-initialYear;
  for (int it=preSimulationYears;it<MTHREAD->MD->getIntSetting("simulationYears")+
preSimulationYears;it++){
    iteration = it;
    year = iteration+MTHREAD->MD->getCachedInitialYear();
    MTHREAD->upgradeMainSBLabel("New year started..");
    msgOut(MSG_INFO, "### "+i2s(getYear())+ " year started.. ####");
    time_t now;
    time(&now);
    struct tm *current = localtime(&now);
    string timemessage = "("+i2s(current->tm_hour)+":"+i2s(current->tm_min)+":"+
i2s(current->tm_sec)+")";
    MTHREAD->upgradeYearSBLabel(iteration+
MTHREAD->MD->getIntSetting("initialYear"));
    MTHREAD->treeViewerChangeGeneralPropertyValue("year",
i2s(iteration+ MTHREAD->MD->getIntSetting("initialYear")));
    if(MTHREAD->MD->getBoolSetting("usePixelData")){
      //MTHREAD->GIS->initLayersModelData(); // removed 20120930, not needed, as data in specific pixel
 values
      MTHREAD->SCORE->runSimulationYear();
    } else {
      MTHREAD->CORE->runSimulationYear();
    }


    //MTHREAD->DO->print(); // done within modelcore now

    for(int i=0;i<MTHREAD->GIS->getXNPixels();i++){
      MTHREAD->GIS->getPixel(i)->newYear(); //delete objects for the pixels, in
 the update the agents will do the same for their objects
    }
  }
}
```

Here is the call graph for this function:


ModelCoreSpatial::resetPixel Values
ModelCoreSpatial::cachePixel ExogenousData
ModelCoreSpatial::update OtherMapData
ModelCoreSpatial::update MapAreas
ModelData::getBoolSetting
BaseClass::s2b
ModelCoreSpatial::runBiological Module
ModelCoreSpatial::runManagement Module
ModelCoreSpatial::sumRegional ForData
ModelData::getBaseData
ModelCoreSpatial::runSimulationYear
ModelCoreSpatial::register CarbonEvents
ModelCoreSpatial::compute Inventary
Output::printDebugPixelValues
BaseClass::i2s
BaseClass::msgOut
ModelCoreSpatial::runMarket Module
ModelCore::runMarketModule
ModelData::getIntSetting
BaseClass::s2i
ModelCoreSpatial::compute CumulativeData
getYear
ModelCore::runManagement Module
Output::print
ModelCore::runSimulationYear
Output::printProductData
run
ModelCore::computeInventary
ThreadManager::upgradeMain SBLabel
ModelCore::computeCumulativeData
ThreadManager::upgradeYear SBLabel
ModelData::getCachedInitialYear
ModelCore::updateMapAreas
ThreadManager::treeViewer ChangeGeneralPropertyValue
Output::printForestData
Gis::getXNPixels
ModelCore::runBiological Module
Gis::getPixel
Pixel::newYear

Here is the caller graph for this function:



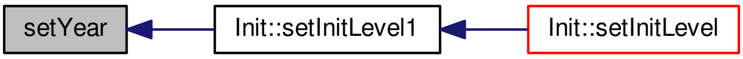
4.39.3.5 int setYear (const int & *year_h*) [inline]

Definition at line 50 of file Scheduler.h.

Referenced by Init::setInitLevel1().

```
00050 {year = year_h;}
```

Here is the caller graph for this function:



4.39.4 Member Data Documentation

4.39.4.1 int iteration [private]

Definition at line 54 of file Scheduler.h.

Referenced by getIteration(), run(), and Scheduler().

4.39.4.2 int year [private]

Definition at line 55 of file Scheduler.h.

Referenced by advanceYear(), getYear(), run(), and setYear().

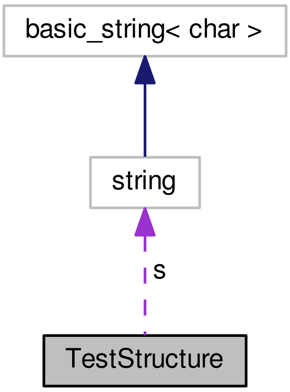
The documentation for this class was generated from the following files:

- /home/lobianco/git/ffsm_pp/src/Scheduler.h
- /home/lobianco/git/ffsm_pp/src/Scheduler.cpp

4.40 TestStructure Struct Reference

```
#include <Sandbox.h>
```

Collaboration diagram for TestStructure:



Public Attributes

- int i
- string s
- double cachedOffer
- double random

4.40.1 Detailed Description

Definition at line 68 of file Sandbox.h.

4.40.2 Member Data Documentation

4.40.2.1 double cachedOffer

Definition at line 72 of file Sandbox.h.

Referenced by testThread::assignJob().

4.40.2.2 int i

Definition at line 70 of file Sandbox.h.

Referenced by Sandbox::testThreads().

4.40.2.3 double random

Definition at line 73 of file Sandbox.h.

Referenced by Sandbox::testThreads().

4.40.2.4 string s

Definition at line 71 of file Sandbox.h.

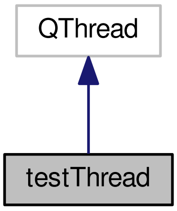
The documentation for this struct was generated from the following file:

- /home/lobianco/git/ffsm_pp/src/Sandbox.h

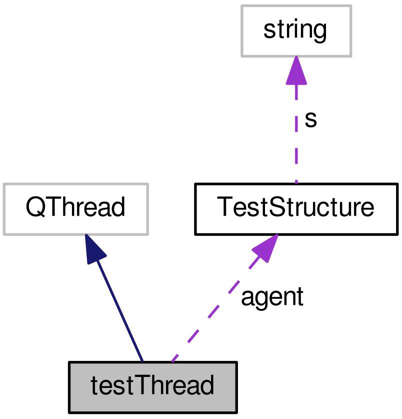
4.41 testThread Class Reference

`#include <Sandbox.h>`

Inheritance diagram for testThread:



Collaboration diagram for testThread:



Public Member Functions

- testThread ()
- void assignJob (TestStructure ∗agent_h)

Protected Member Functions

- void run ()

Private Attributes

- volatile TestStructure ∗ agent

4.41.1 Detailed Description

Definition at line 77 of file Sandbox.h.

4.41.2 Constructor & Destructor Documentation

4.41.2.1 testThread ( )

Definition at line 1476 of file Sandbox.cpp.

```
01476                           {
01477 
01478 }
```

4.41.3 Member Function Documentation

4.41.3.1 void assignJob (TestStructure ∗ *agent_h*)

Definition at line 1496 of file Sandbox.cpp.

```
01496                                              {
01497   agent = agent_h;
01498   agent->cachedOffer = 0;
01499 }
```

4.41.3.2 void run ( ) [protected]

Definition at line 1481 of file Sandbox.cpp.

```
01481                  {
01482 
01483   cout << agent->i << endl;
01484 
01485   double randChange =  (0+( (double)rand() / ((double)(RAND_MAX)+(double)(1)) )*(10-0+1))/ (double)100; //
      rand() must be not thread safe !!!!
01486 
01487   int justn = 10000;
01488   vector <double> takeTimeVector (justn, 0);
01489   for (int i =0; i< justn;i++){
01490     takeTimeVector.at(i)=i*2;
01491   }
01492   agent->cachedOffer = agent->random;
01493 }
```

4.41.4 Member Data Documentation

4.41.4.1 volatile **TestStructure**∗ agent `[private]`

Definition at line 88 of file Sandbox.h.

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

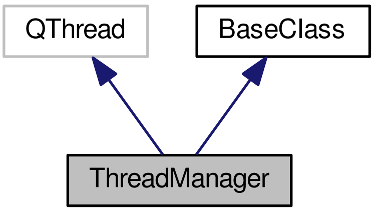
- /home/lobianco/git/ffsm_pp/src/Sandbox.h
- /home/lobianco/git/ffsm_pp/src/Sandbox.cpp

4.42 ThreadManager Class Reference

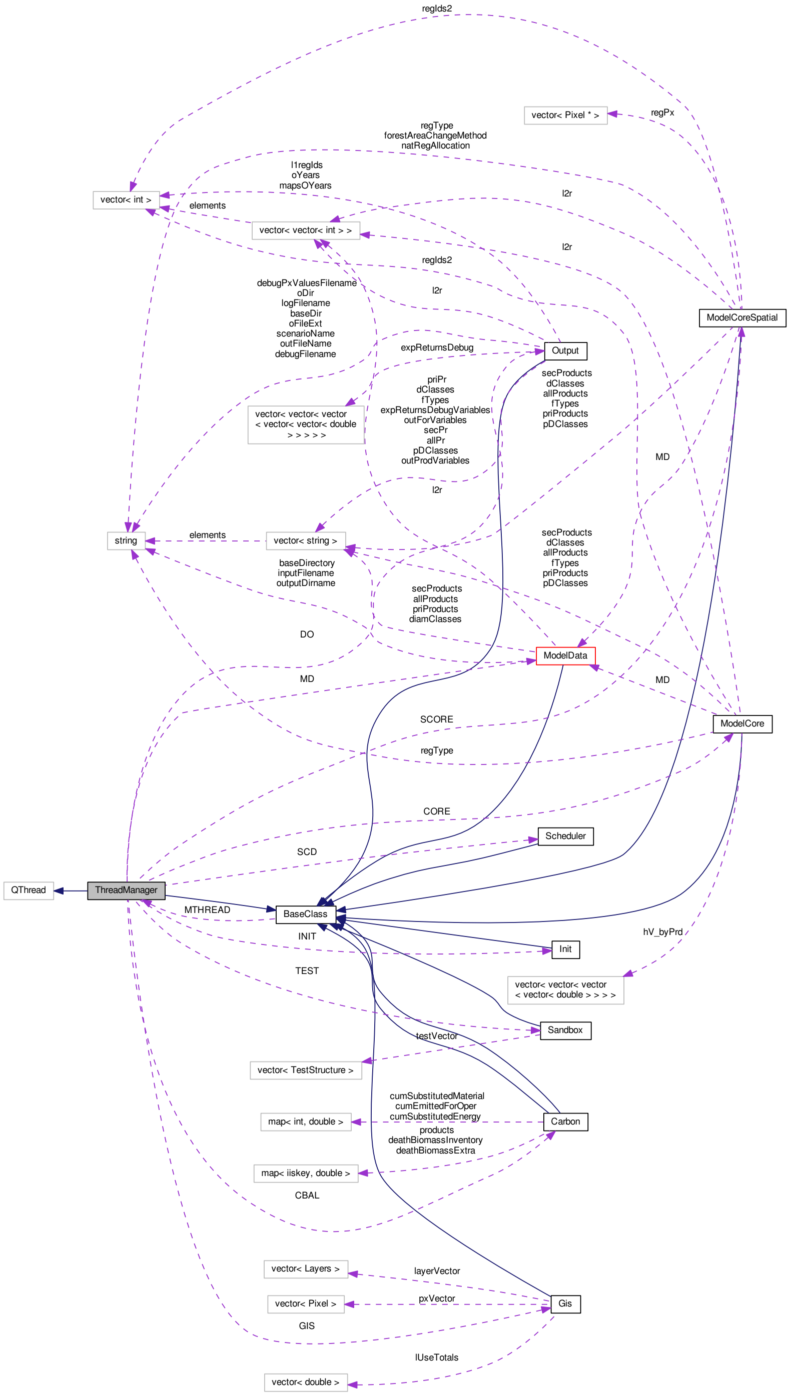
Thread manager. Responsable to manage the main thread and "speak" with the GUI.

`#include <ThreadManager.h>`

Inheritance diagram for ThreadManager:



Collaboration diagram for ThreadManager:



Public Slots

- void checkQuery (int px_ID, int currentLayerIndex, bool newRequest=true)

  Switch and control the access to pxQueryID and layerQueryPos members.
- void computeQuery (int px_ID, int currentLayerIndex)

  Compute the pixel query and return it to the GUI (with a signal)
- void retrieveScenarioNameFromGUI (const QString &scenarioName_h)

Signals

- void upgradeLogArea (const QString &logMessage)
- void upgradeMainSBLabelToGui (const QString &logMessage)
- void upgradeYearSBLabelToGui (const QString &logMessage)
- void addLayerToGui (QString layerName, QString layerLabel)
- void updatePixelToGui (QString layerName_h, int x_h, int y_h, QColor color)
- void updateImageToGui (QString layerName_h, QImage image_h)
- void setOutputDirNameToGui (string outputDirname_h)
- void setGUIUnsavedStatus (bool status_h)
- void setGUIMapDimension (int x_h, int y_h)
- void treeViewerItemChangeValueToGui (string itemID, string newValue)
- void treeViewerItemRemoveToGui (string itemID)
- void treeViewerAddItemToGui (string text, string itemID, string parentID)
- void fitInWindowToGui ()
- void queryRequestOnPx (int px_ID, int currentLayerIndex)
- void publishQueryResults (const QString &results)
- void activateTab (int pos_h)
- void resetGUIForNewSimulation ()
- void sendScenarioOptionsToGUI (const QVector$<$ QString $>$ &scenarios_h)

Public Member Functions

- ThreadManager ()
- void setMessage (const QString &message)
- void stop ()
- void deleteDeadOldPointers ()

  Useful for several model running without leaving the GUI.
- void pauseOrResume ()
- void pause ()
- void resume ()
- void refreshGUI ()
- void msgOut (const int msgCode_h, const string message_h)
- void addLayer (string layerName_h, string layerLabel_h)
- void updatePixel (string layerName_h, int x_h, int y_h, QColor color)
- void updateImage (string layerName_h, const QImage &image_h)
- void upgradeMainSBLabel (const string message_h)
- void upgradeYearSBLabel (int year)
- string getBaseDirectory ()
- string getInputFileName ()
- string getScenarioName ()
- void setScenarioName (const string &scenarioName_h)
- void setOutputDirName (string outputDirname_h)
- void setMDPointer (ModelData ∗MD_h)

  the regional data object..
- void setGISPointer (Gis ∗GIS_h)

  GIS information and methods..
- void setINITPointer (Init ∗INIT_h)

  the Init object, it schedule the pre-simulation phase..
- void setTestPointer (Sandbox ∗TEST_h)

  the sandbox object for within-development quick tests
- void setSCDPointer (Scheduler ∗SCD_h)

  the scheduler object. It manage the simulation loops..

- void setDOPointer (Output ∗DO_h)

  manage the printing of data needed for scenario-analisys. The "message output" (needed to see "what is it happening?" are instead simply printed with msgOut()..
- void setCOREPointer (ModelCore ∗CORE_h)

  Perform the algorithms of the model.
- void setSCOREPointer (ModelCoreSpatial ∗SCORE_h)

  Perform the algorithms of the model.
- void setOPTPointer (Ipopt::SmartPtr< Ipopt::TNLP > OPT_h)

  Perform the market optimisation.
- void setCBALPointer (Carbon ∗CBAL_h)

  Module that account for the Carbon Balance.
- void setInputFileName (QString inputFileName_h)
- void treeViewerChangeGeneralPropertyValue (string propertyName, string newValue)
- void fitInWindow ()
- void runFromConsole (QString inputFileName_h, QString scenarioName_h)

  Re-draw the map making it to fit (with the right proportions) to the widget.
- bool usingGUI ()

Public Attributes

- ModelData ∗ MD

  the model data object
- Gis ∗ GIS

  GIS information and methods.
- Init ∗ INIT

  the Init object (pre-simulation scheduler)
- Scheduler ∗ SCD

  the scheduler object (simulation-loops scheduler)
- Output ∗ DO

  data output
- ModelCore ∗ CORE

  Core of the model.
- ModelCoreSpatial ∗ SCORE

  Core of the model (spatial version)
- Carbon ∗ CBAL

  Module for the Carbon Balance.
- Sandbox ∗ TEST

  Various debugging code for development.
- Ipopt::SmartPtr< Ipopt::TNLP > OPT

  Market optimisation.
- std::mt19937 ∗ gen

  used in the sampling from normal distribution

Protected Member Functions

- void run ()

Private Attributes

- QString messageStr
- volatile bool stopped
- volatile bool running
- QString inputFileName
- QString baseDirectory
- QString scenarioName
- volatile int pxQueryID
- volatile int layerQueryPos
- QMutex mutex
- bool GUI

Additional Inherited Members

4.42.1 Detailed Description

Thread manager. Responsable to manage the main thread and "speak" with the GUI.

ThreadManager is responsable for the actions on the main thread (run/pause/resume/stop) and to speack with the GUI using the signal/slot tecniques.

Author

Antonello Lobianco

Definition at line 65 of file ThreadManager.h.

4.42.2 Constructor & Destructor Documentation

4.42.2.1 ThreadManager ( )

Definition at line 35 of file ThreadManager.cpp.

```
00035                               {
00036   running=false;
00037   stopped=false;
00038   layerQueryPos = -1;
00039
00040   // initializing pointers...
00041   MD      = NULL;
00042   GIS     = NULL;
00043   INIT    = NULL;
00044   SCD     = NULL;
00045   DO      = NULL;
00046   CORE    = NULL;
00047   SCORE   = NULL;
00048   TEST    = NULL;
00049   CBAL    = NULL;
00050   //randev = NULL;
00051   gen     = NULL;
00052
00053   GUI = false;
00054
00055   scenarioName="";
00056   inputFileName="";
00057   baseDirectory="";
00058
00059 }
```

4.42.3 Member Function Documentation

4.42.3.1 void activateTab (int *pos_h*) [signal]

4.42.3.2 void addLayer (string *layerName_h,* string *layerLabel_h*)

Definition at line 251 of file ThreadManager.cpp.

```
00251                                                                  {
00252   QString layerName = layerName_h.c_str();
00253   QString layerLabel = layerLabel_h.c_str();
00254   emit addLayerToGui(layerName, layerLabel);
00255 }
```

4.42.3.3 void addLayerToGui (QString *layerName,* QString *layerLabel*) [signal]

4.42.3.4 void checkQuery (int *px_ID,* int *currentLayerIndex,* bool *newRequest* = true) [slot]

Switch and control the access to pxQueryID and layerQueryPos members.

checkQuery() is a function that can be called my the GUI trough a signal or from the running thread under refreshGUI(), and it is protected with a mutex.
It's role is to control the status of pxQueryID and layerQueryPos member variables.
If the call come from the GUI, it is a new request and we set them to the new values, otherwise we gonna see if they are just beed changed and if so (layerQueryPos>=0) we call computeQuery().

Definition at line 285 of file ThreadManager.cpp.

```
00285                                                                              {
00286   QMutexLocker locker(&mutex);
00287   if(newRequest){
00288     pxQueryID = px_ID;
00289     layerQueryPos = currentLayerIndex;
00290     if(stopped){computeQuery(pxQueryID,
      layerQueryPos);layerQueryPos = -1;} // model is stopped, no way the model thread
       will do the query work
00291     else{emit publishQueryResults("<i>..wait.. processing query..</i>");} // model is
       running.. it will be the model thread to execute the query
00292     return;
00293   } else {
00294     if(layerQueryPos<0){
00295       return;
00296     } else {
00297       computeQuery(pxQueryID, layerQueryPos);
00298       layerQueryPos = -1;
00299       return;
00300     }
00301   }
00302 }
```

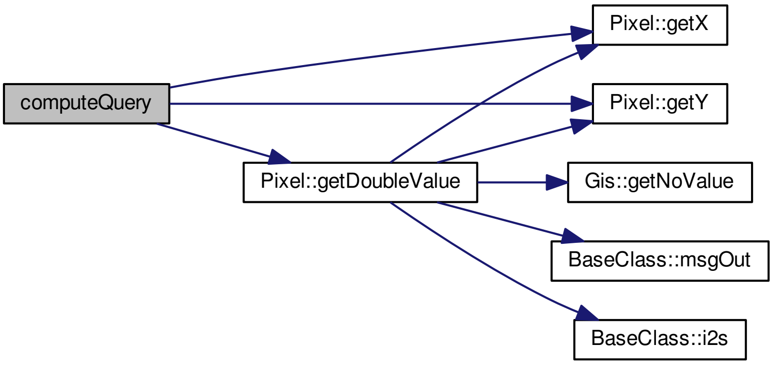
4.42.3.5 void computeQuery (int *px_ID,* int *currentLayerIndex*) [slot]

Compute the pixel query and return it to the GUI (with a signal)

Definition at line 305 of file ThreadManager.cpp.

```
                              {

 // IMPORTANT: this function is called at refreshGUI() times, so if there are output messages, call them with the option to NOT refresh the gui, otherwise we go to an infinite loop...

 vector<Layers*> layers;
 try {
   layers = GIS->getLayerPointers();
 }catch (...) {
   emit activateTab(2); // tell the gui to activate the 3rd page, those with the pixel info
   emit publishQueryResults("GIS pointer is dead.. maybe simulation has ended???");
   return;
 }
 QString result= "";
 int realID = GIS->sub2realID(px_ID);
 if (realID<0) {
   emit publishQueryResults("Query result: Spatial data is not yet ready in the model. Please click again later.");
   return; // on early stage we may have errors, and here we prevent this error to have further consequences.
 }
 Pixel* px;
 try {
   px = GIS->getPixel(realID);
 }catch (...) {
   emit activateTab(2); // tell the gui to activate the 3rd page, those with the pixel info
   emit publishQueryResults("Query result: Spatial data is not yet ready in the model. Please click again later.");
   return;
 }
 result += "Pixel: ";
 result += i2s(realID).c_str();
 result += " (";
 result += i2s(px->getX()).c_str();
 result += ",";
 result += i2s(px->getY()).c_str();
 result += ")";
 result +="<p><table>";
 uint countVisibleLayers = 0;
 for (uint i=0;i<layers.size();i++){
   if(!layers[i]->getDisplay()){
     continue;
   }
   QString boldStart="";
   QString boldEnd = "";
   if (countVisibleLayers == currentLayerIndex){
     boldStart = "<b>";
     boldEnd   = "</b>";
   }
   result += "<tr>";
   string layerName = layers[i]->getName();
   double value = px->getDoubleValue(layerName);
   string category = layers[i]->getCategory(value);
   //QColor color = layers[i]->getColor(value);
   result += "<td>";
   result += boldStart;
   result += layerName.c_str();
   result += boldEnd;
   result += "</td><td>";
   result += boldStart;
   result += category.c_str();
   result += boldEnd;
   result += "</td>";
   result += "</tr>";
   if(layers[i]->getDisplay()){ // if not really needed, but ok if we decide to change and get displayed also hidden layers
     countVisibleLayers++;
   }
 }
 result += "</table>";
 emit activateTab(2); // tell the gui to activate the 3rd page, those with the pixel info
 emit publishQueryResults(result);
}
```

Here is the call graph for this function:



4.42.3.6 void deleteDeadOldPointers ( )

Useful for several model running without leaving the GUI.

Delete the pointers (e.g. GIS) eventually remained from a previous run.
This function is called at the START of a new simulation, and it will check if model pointers (e.g. GIS) exist , and if so it will delete them.
This is useful when we keep the MainWindow open but we run the model for a second time.
Why we don't delete them at the end of a simulation, instead of deleting them on a new run? That's because we want let the user to interface with the model even when this is ended, w.g. for query the map.

Definition at line 157 of file ThreadManager.cpp.

```
00157                                          {
00158   if (DO) {delete DO; DO=0;}
00159   if (INIT) {delete INIT; INIT=0;}
00160   if (SCD) {delete SCD; SCD=0;}
00161   if (GIS) {delete GIS; GIS=0;}
00162   if (MD) {delete MD; MD=0;}
00163   if (CORE){delete CORE; CORE=0;}
00164   if (SCORE){delete SCORE; SCORE=0;}
00165   if (CBAL) {delete CBAL; CBAL=0;}
00166   //if (OPT) {delete OPT; OPT=0;} // not needed, it's a "smart point"
00167   if(TEST){delete TEST; TEST=0;}
00168   //if(randev){delete randev; randev=0;}
00169   if(gen){delete gen; gen=0;}
00170 }
```

4.42.3.7 void fitInWindow ( ) [inline]

Definition at line 148 of file ThreadManager.h.

```
00148 {emit fitInWindowToGui();}; ///< Re-draw the map making it to fit (with the right
      proportions) to the widget
```

4.42.3.8 void fitInWindowToGui ( ) [signal]

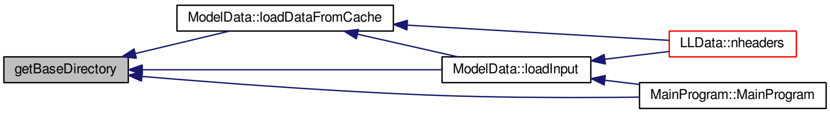
4.42.3.9 string getBaseDirectory ( ) [inline]

Definition at line 98 of file ThreadManager.h.

Referenced by ModelData::loadDataFromCache(), ModelData::loadInput(), and MainProgram::MainProgram().

```
00098 {return baseDirectory.toStdString();};
```

Here is the caller graph for this function:



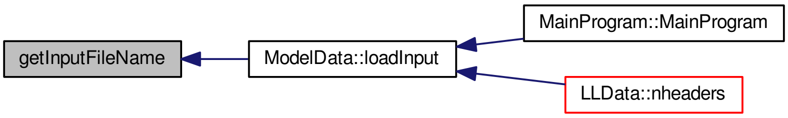
4.42.3.10 string getInputFileName ( ) [inline]

Definition at line 99 of file ThreadManager.h.

Referenced by ModelData::loadInput().

```
00099 {return inputFileName.toStdString();};
```

Here is the caller graph for this function:



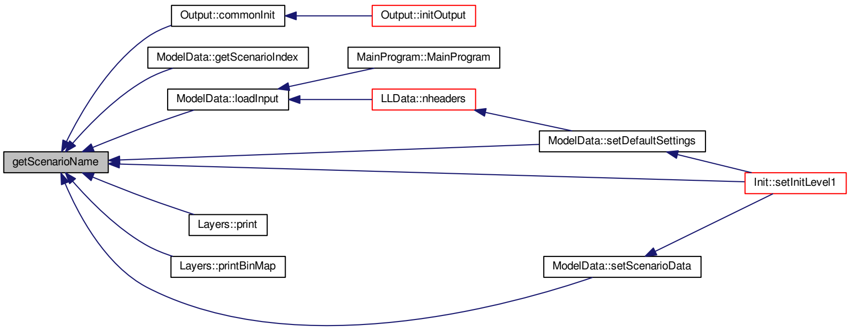
4.42.3.11 string getScenarioName ( ) [inline]

Definition at line 100 of file ThreadManager.h.

Referenced by Output::commonInit(), ModelData::getScenarioIndex(), ModelData::loadInput(), Layers::print(), Layers::printBinMap(), ModelData::setDefaultSettings(), Init::setInitLevel1(), and ModelData::setScenarioData().

```
00100 {return scenarioName.toStdString();};
```

Here is the caller graph for this function:



4.42.3.12 void msgOut (const int *msgCode_h,* const string *message_h*)

Definition at line 237 of file ThreadManager.cpp.

```
00237                                                              {
00238   QString message = message_h.c_str();
00239   emit upgradeLogArea(message);
00240   if (msgCode_h == 2){
00241     emit upgradeMainSBLabelToGui(message);
00242   }
00243 }
```

4.42.3.13 void pause ( )

Definition at line 195 of file ThreadManager.cpp.

```
00195                       {
00196   if(!stopped){
00197     if(running){
00198       running= false;
00199     }
00200     else {
00201       return;
00202     }
00203   }
00204   return;
00205 }
```

4.42.3.14 void pauseOrResume ( )

Definition at line 179 of file ThreadManager.cpp.

```
00179                              {
00180   if(!stopped){
00181     if(running){
00182       running= false;
00183       emit upgradeLogArea("PAUSE clicked PAUSING");
00184     }
00185     else {
00186       running=true;
00187       emit upgradeLogArea("PAUSE clicked RESUMING");
00188       emit setGUIUnsavedStatus(true);
00189     }
00190   }
00191   return;
00192 }
```

4.42.3.15 void publishQueryResults (const QString & *results*) `[signal]`

4.42.3.16 void queryRequestOnPx (int *px_ID*, int *currentLayerIndex*) `[signal]`

4.42.3.17 void refreshGUI ( )

Definition at line 222 of file ThreadManager.cpp.

```
00222                           {
00223     checkQuery(0,0,false);
00224     while (!running){
00225       if(stopped){
00226         break;
00227       }
00228     }
00229     if (stopped){
00230       emit upgradeLogArea("Model has been stopped.");
00231       running= false;
00232       throw(2);
00233     }
00234 }
```

4.42.3.18 void resetGUIForNewSimulation ( ) `[signal]`

4.42.3.19 void resume ( )

Definition at line 208 of file ThreadManager.cpp.

```
00208                         {
00209   if(!stopped){
00210     if(running){
00211       return;
00212     }
00213     else {
00214       running=true;
00215       emit setGUIUnsavedStatus(true);
00216     }
00217   }
00218   return;
00219 }
```

4.42.3.20 void retrieveScenarioNameFromGUI (const QString & *scenarioName_h*) [slot]

Definition at line 113 of file ThreadManager.cpp.

```
00113                                                                              {
00114   scenarioName = scenarioName_h;
00115   msgOut(MSG_INFO, "Selected scenario: "+scenarioName.toStdString());
00116   cout << "Selected scenario: "+scenarioName.toStdString() << endl;
00117   resume();
00118 }
```

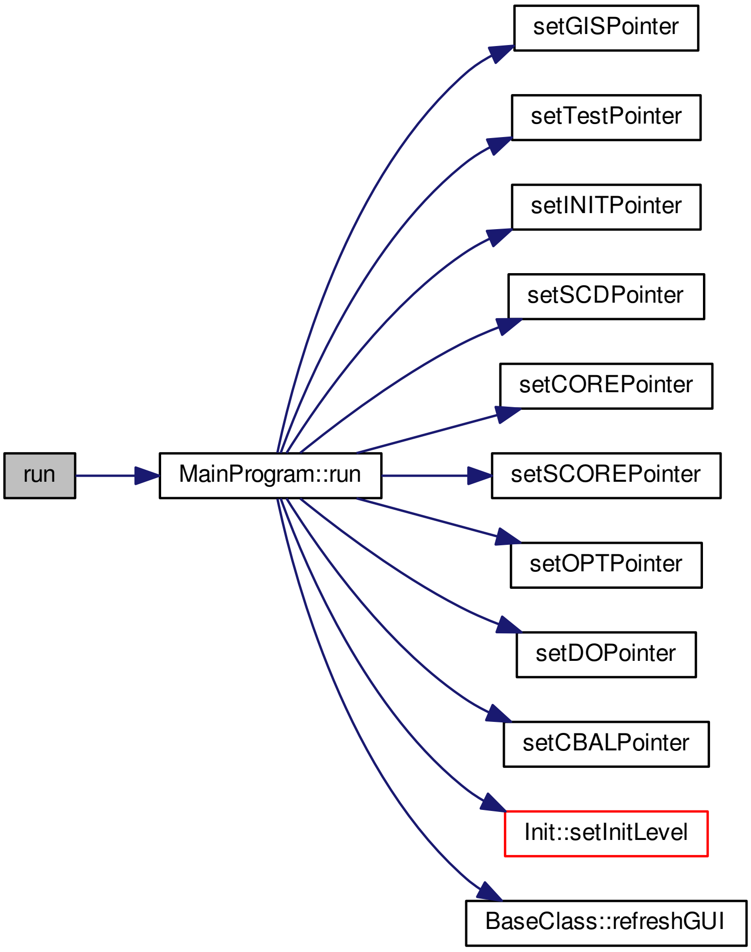
4.42.3.21 void run ( ) [protected]

Todo .. perform a better exception handing..

Definition at line 66 of file ThreadManager.cpp.

```
00066                     {
00067   running=true;
00068   stopped=false;
00069 
00070   srand(1);
00071   GUI=true;
00072 
00073   emit upgradeLogArea("**INFO: Start running the model...");
00074 
00075   MainProgram* myProgram;
00076   try{
00077     deleteDeadOldPointers();
00078     emit resetGUIForNewSimulation();
00079 
00080 
00081     QFileInfo file(inputFileName);
00082     QDir baseDir = file.absoluteDir();
00083     baseDirectory = baseDir.absolutePath()+"/";
00084     myProgram = new MainProgram(this);
00085 
00086     //myProgram->setBaseDirectory(baseDirectory);
00087 
00088     vector<string> scenarios = MD->getScenarios();
00089     QVector<QString> qscenarios;
00090     for(uint i=0;i<scenarios.size();i++){
00091       qscenarios.push_back(scenarios.at(i).c_str());
00092     }
00093     running = false;
00094     emit sendScenarioOptionsToGUI(qscenarios);
00095     refreshGUI();
00096 
00097     myProgram->run();
00098 
00099     // Here the model has come to an end...
00100     running=false;
00101     stopped=true;
00102     delete myProgram;
00103     refreshGUI();
00104 
00105   }catch (...) { /// \todo .. perform a better exception handing..
00106     emit upgradeLogArea("**INFO: Model has stopped or rised an error (read previous line).");
00107   }
00108   emit upgradeLogArea("**INFO: Model has ended.");
00109 
00110 }
```

Here is the call graph for this function:



4.42.3.22 void runFromConsole (QString *inputFileName_h,* QString *scenarioName_h*)

Re-draw the map making it to fit (with the right proportions) to the widget.

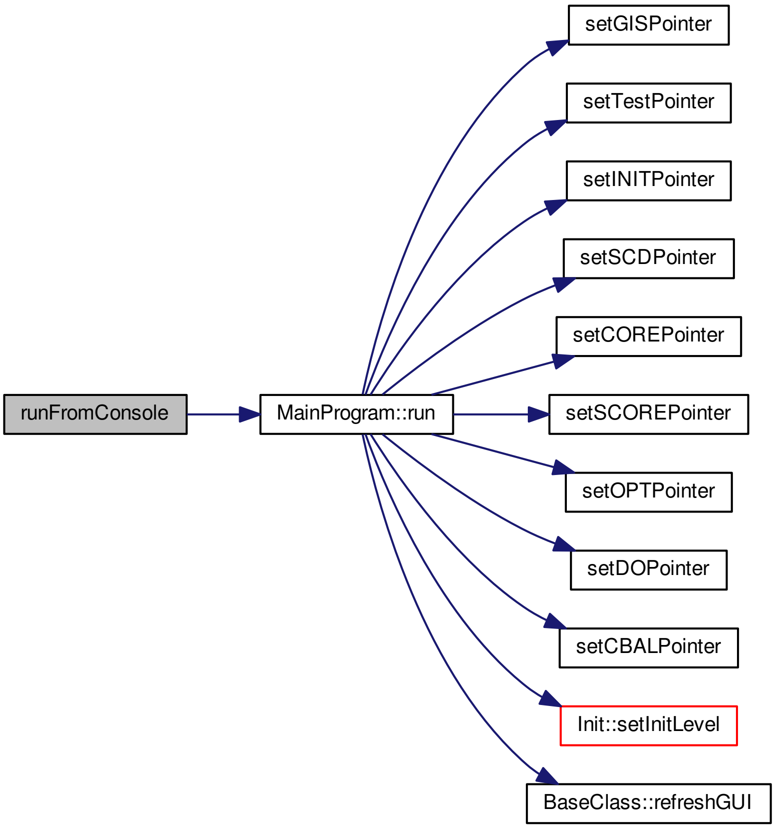
Definition at line 121 of file ThreadManager.cpp.

Referenced by main().

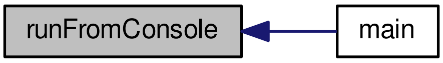
```
                                                                        {
          GUI = false;
          scenarioName = scenarioName_h;
          inputFileName = inputFileName_h;
          QFileInfo file(inputFileName);
          QDir baseDir = file.absoluteDir();
          baseDirectory = baseDir.absolutePath()+"/";
          cout <<"Using base directory: "<< baseDirectory.toStdString() << endl;


          MainProgram* myProgram = new MainProgram(this);

          if( scenarioName_h == ""){ // if the scenario option has not been choosed, go for the first one!
```

```
      vector<string> scenarios = MD->getScenarios();
      scenarioName = scenarios.at(0).c_str();
    }

    //myProgram->setBaseDirectory(baseDirectory);
    myProgram->run();
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.42.3.23 void sendScenarioOptionsToGUI (const QVector< QString > & *scenarios_h*) [signal]

4.42.3.24 void setCBALPointer (**Carbon** ∗ *CBAL_h*) `[inline]`

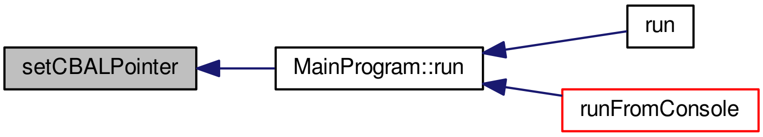
Module that account for the Carbon Balance.

Definition at line 123 of file ThreadManager.h.

Referenced by MainProgram::run().

```
00123 {CBAL=CBAL_h;};
```

Here is the caller graph for this function:



4.42.3.25 void setCOREPointer (**ModelCore** ∗ *CORE_h*) `[inline]`

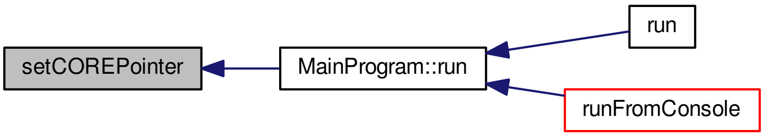
Perform the algorithms of the model.

Definition at line 117 of file ThreadManager.h.

Referenced by MainProgram::run().

```
00117 {CORE=CORE_h;};
```

Here is the caller graph for this function:



4.42.3.26 void setDOPointer (**Output** ∗ *DO_h*) [inline]

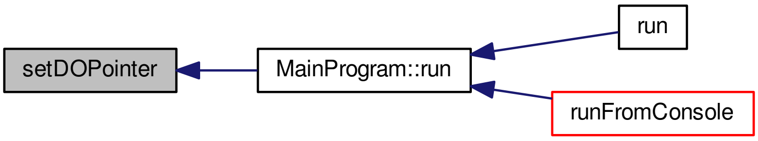
manage the printing of data needed for scenario-analisys. The "message output" (needed to see "what is it happening?" are instead simply printed with msgOut()..

Definition at line 115 of file ThreadManager.h.

Referenced by MainProgram::run().

```
00115 {DO=DO_h;};
```

Here is the caller graph for this function:



4.42.3.27 void setGISPointer (**Gis** ∗ *GIS_h*) [inline]

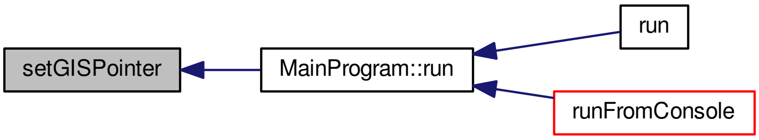
GIS information and methods..

Definition at line 107 of file ThreadManager.h.

Referenced by MainProgram::run().

```
00107 {GIS=GIS_h;};
```

Here is the caller graph for this function:



4.42.3.28 void setGUIMapDimension (int *x_h,* int *y_h*) `[signal]`

4.42.3.29 void setGUIUnsavedStatus (bool *status_h*) `[signal]`

4.42.3.30 void setINITPointer (Init ∗ *INIT_h*) `[inline]`

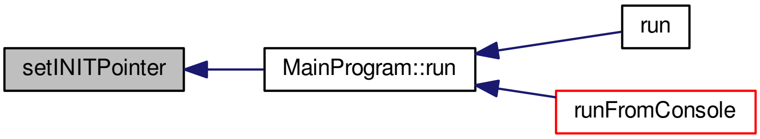
the Init object, it schedule the pre-simulation phase..

Definition at line 109 of file ThreadManager.h.

Referenced by MainProgram::run().

```
00109 {INIT=INIT_h;};
```

Here is the caller graph for this function:



4.42.3.31 void setInputFileName (QString *inputFileName_h*)

Definition at line 143 of file ThreadManager.cpp.

```
00143                                                        {
00144   inputFileName= inputFileName_h;
00145   QFileInfo file(inputFileName);
00146   QDir baseDir = file.absoluteDir();
00147   baseDirectory = baseDir.absolutePath()+"/";
00148 }
```

4.42.3.32 void setMDPointer (ModelData ∗ *MD_h*) `[inline]`

the regional data object..

Definition at line 105 of file ThreadManager.h.

Referenced by MainProgram::MainProgram().

```
00105 {MD=MD_h;};
```

Here is the caller graph for this function:

setMDPointer
MainProgram::MainProgram

4.42.3.33 void setMessage (const QString & *message*)

Definition at line 62 of file ThreadManager.cpp.

```
00062                                              {
00063     messageStr = message;
00064 }
```

4.42.3.34 void setOPTPointer (Ipopt::SmartPtr< Ipopt::TNLP > *OPT_h*) [inline]

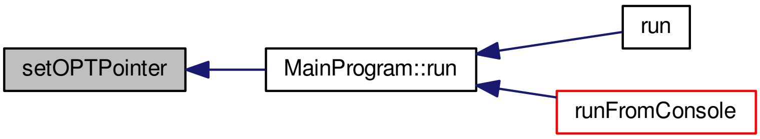
Perform the market optimisation.

Definition at line 121 of file ThreadManager.h.

Referenced by MainProgram::run().

```
00121 {OPT=OPT_h;};
```

Here is the caller graph for this function:



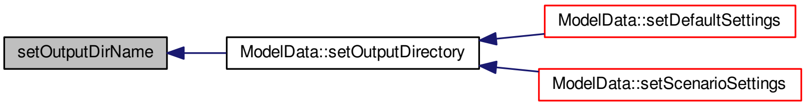
4.42.3.35 void setOutputDirName (string *outputDirname_h*)

Definition at line 246 of file ThreadManager.cpp.

Referenced by ModelData::setOutputDirectory().

```
00246                                                     {
00247   emit setOutputDirNameToGui(outputDirname_h);
00248 }
```

Here is the caller graph for this function:


setOutputDirName
ModelData::setOutputDirectory
ModelData::setDefaultSettings
ModelData::setScenarioSettings

4.42.3.36 void setOutputDirNameToGui (string *outputDirname_h*) [signal]

4.42.3.37 void setSCDPointer (Scheduler ∗ *SCD_h*) [inline]

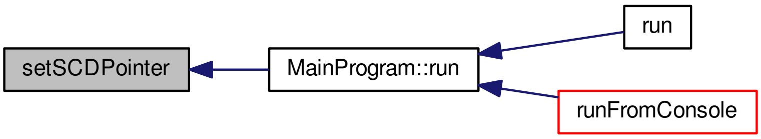
the scheduler object. It manage the simulation loops..

Definition at line 113 of file ThreadManager.h.

Referenced by MainProgram::run().

```
00113 {SCD=SCD_h;};
```

Here is the caller graph for this function:



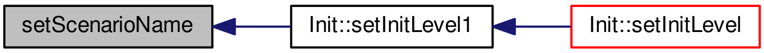
4.42.3.38 void setScenarioName (const string & *scenarioName_h*) [inline]

Definition at line 101 of file ThreadManager.h.

Referenced by Init::setInitLevel1().

```
00101 {scenarioName=scenarioName_h.c_str();};
```

Here is the caller graph for this function:



4.42.3.39 void setSCOREPointer (**ModelCoreSpatial** ∗ *SCORE_h*) [inline]

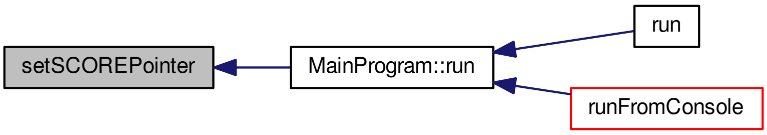
Perform the algorithms of the model.

Definition at line 119 of file ThreadManager.h.

Referenced by MainProgram::run().

```
00119 {SCORE=SCORE_h;};
```

Here is the caller graph for this function:



4.42.3.40 void setTestPointer (**Sandbox** ∗ *TEST_h*) [inline]

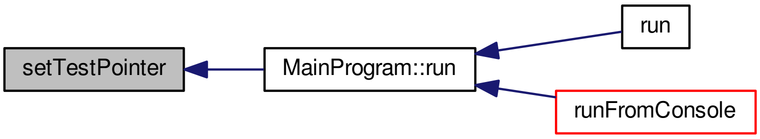
the sandbox object for within-development quick tests

Definition at line 111 of file ThreadManager.h.

Referenced by MainProgram::run().

```
00111 {TEST=TEST_h;};
```

Here is the caller graph for this function:



4.42.3.41 void stop ( )

Definition at line 173 of file ThreadManager.cpp.

```
00173                     {
00174     stopped = true;
00175   emit upgradeLogArea("STOP cliccked stopping");
00176 }
```

4.42.3.42 void treeViewerAddItemToGui (string *text*, string *itemID*, string *parentID*) [signal]

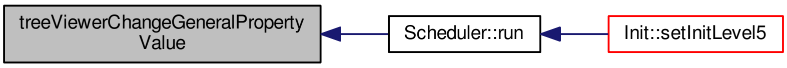
4.42.3.43 void treeViewerChangeGeneralPropertyValue (string *propertyName*, string *newValue*) [inline]

Definition at line 144 of file ThreadManager.h.

Referenced by Scheduler::run().

```
00144                                                                                          {
00145                            emit treeViewerItemChangeValueToGui("general_"+
      propertyName, newValue);};
```

Here is the caller graph for this function:



4.42.3.44 void treeViewerItemChangeValueToGui (string *itemID*, string *newValue*) [signal]

4.42.3.45 void treeViewerItemRemoveToGui (string *itemID*) [signal]

4.42.3.46 void updateImage (string *layerName_h*, const QImage & *image_h*)

Definition at line 263 of file ThreadManager.cpp.

```
00263                                                                    {
00264   emit updateImageToGui(layerName_h.c_str(), image_h);
00265 }
```

4.42.3.47 void updateImageToGui (QString *layerName_h*, QImage *image_h*) [signal]

4.42.3.48 void updatePixel (string *layerName_h*, int *x_h*, int *y_h*, QColor *color*)

Definition at line 258 of file ThreadManager.cpp.

```
00258                                                                             {
00259   emit updatePixelToGui(layerName_h.c_str(), x_h, y_h, color_h);
00260 }
```

4.42.3.49 void updatePixelToGui (QString *layerName_h,* int *x_h,* int *y_h,* QColor *color*) [signal]

4.42.3.50 void upgradeLogArea (const QString & *logMessage*) [signal]

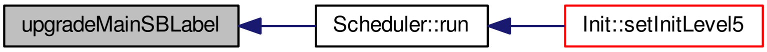
4.42.3.51 void upgradeMainSBLabel (const string *message_h*)

Definition at line 268 of file ThreadManager.cpp.

Referenced by Scheduler::run().

```
00268                                                      {
00269   emit upgradeMainSBLabelToGui(message_h.c_str());
00270 }
```

Here is the caller graph for this function:



4.42.3.52 void upgradeMainSBLabelToGui (const QString & *logMessage*) [signal]

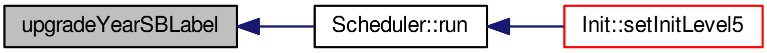
4.42.3.53 void upgradeYearSBLabel (int *year*)

Definition at line 273 of file ThreadManager.cpp.

Referenced by Scheduler::run().

```
00273                                         {
00274   QString temp;
00275   temp= i2s(year).c_str();
00276   emit upgradeYearSBLabelToGui(temp);
00277 }
```

Here is the caller graph for this function:



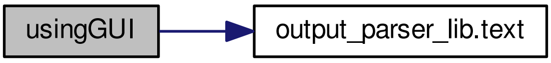
4.42.3.54 void upgradeYearSBLabelToGui (const QString & *logMessage*) [signal]

4.42.3.55 bool usingGUI ( ) [inline]

Definition at line 150 of file ThreadManager.h.

```
00150 {return GUI;};
```

Here is the call graph for this function:



4.42.4 Member Data Documentation

4.42.4.1 QString baseDirectory [private]

Definition at line 188 of file ThreadManager.h.

4.42.4.2 Carbon∗ CBAL

Module for the Carbon Balance.

Definition at line 79 of file ThreadManager.h.

Referenced by ModelCoreSpatial::initialiseCarbonModule(), Output::printCarbonBalance(), and ModelCoreSpatial::registerCarbonEvents().

4.42.4.3 ModelCore∗ CORE

Core of the model.

Definition at line 77 of file ThreadManager.h.

Referenced by Scheduler::run(), and Init::setInitLevel3().

4.42.4.4 Output∗ DO

data output

Definition at line 76 of file ThreadManager.h.

Referenced by ModelCore::runInitPeriod(), ModelCoreSpatial::runInitPeriod(), ModelCore::runManagementModule(), ModelCore::runMarketModule(), ModelCoreSpatial::runMarketModule(), ModelCore::runSimulationYear(), ModelCoreSpatial::runSimulationYear(), Init::setInitLevel3(), and Init::setInitLevel6().

4.42.4.5 std::mt19937∗ gen

used in the sampling from normal distribution

Definition at line 83 of file ThreadManager.h.

Referenced by Init::setInitLevel1().

4.42.4.6 Gis∗ GIS

GIS information and methods.

Definition at line 73 of file ThreadManager.h.

Referenced by ModelCoreSpatial::assignSpMultiplierPropToVols(), Pixel::changeValue(), Layers::countMyPixels(), Layers::getCategory(), Layers::getColor(), Pixel::getDoubleValue(), Pixel::getPathMortality(), Pixel::getPixelsAtDistLevel(), ModelCoreSpatial::initializePixelArea(), ModelCoreSpatial::initializePixelVolumes(), ModelCoreSpatial::loadExogenousForestLayers(), Layers::print(), Layers::printBinMap(), Output::printMaps(), Layers::randomShuffle(), Scheduler::run(), ModelCoreSpatial::runInitPeriod(), Init::setInitLevel1(), ModelRegion::setMyPixels(), ModelCore::updateMapAreas(), ModelCoreSpatial::updateMapAreas(), and ModelCoreSpatial::updateOtherMapData().

4.42.4.7 bool GUI `[private]`

Definition at line 193 of file ThreadManager.h.

4.42.4.8 Init∗ INIT

the Init object (pre-simulation scheduler)

Definition at line 74 of file ThreadManager.h.

Referenced by MainProgram::run().

4.42.4.9 QString inputFileName `[private]`

Definition at line 187 of file ThreadManager.h.

4.42.4.10 volatile int layerQueryPos `[private]`

Definition at line 191 of file ThreadManager.h.

4.42.4.11 ModelData∗ MD

the model data object

Definition at line 72 of file ThreadManager.h.

Referenced by ModelCoreSpatial::assignSpMultiplierPropToVols(), ModelCoreSpatial::cachePixelExogenousData(), ModelCore::cacheSettings(), ModelCoreSpatial::cacheSettings(), Output::commonInit(), ModelCoreSpatial::computeCumulativeData(), ModelCoreSpatial::computeInventary(), ModelRegion::getArea(), ModelData::getAvailableAliveTimber(), ModelData::getBoolSetting(), ModelData::getBoolVectorSetting(), ModelData::getDoubleSetting(), ModelData::getDoubleVectorSetting(), ModelData::getIntSetting(), ModelData::getIntVectorSetting(), Pixel::getMultiplier(), Output::getOutputFieldDelimiter(), Pixel::getPathMortality(), ModelData::getRegionIds(), Pixel::getSpModifier(), Carbon::getStock(), ModelData::getStringSetting(), ModelData::getStringVectorSetting(), ModelCore::gfd(), ModelCoreSpatial::gfd(), ModelCore::gpd(), ModelCoreSpatial::gpd(), Carbon::HWP_eol2energy(), Carbon::initialiseDeathBiomassStocks(), Carbon::initialiseEmissionCounters(), Carbon::initialiseProductsStocks(), ModelCoreSpatial::loadExogenousForestLayers(), MainProgram::MainProgram(), ModelRegion::ModelRegion(), Pixel::Pixel(), Output::print(), Layers::print(), Layers::printBinMap(), Output::printCarbonBalance(), Output::printDebugOutput(), Output::printDebugPixelValues(), Output::printFinalOutput(), Output::printForestData(), Output::printProductData(), ModelData::regId2RegSName(), Carbon::registerDeathBiomass(), Carbon::registerHarvesting(), Carbon::registerProducts(), Carbon::registerTransports(), ModelData::regSName2RegId(), ModelCoreSpatial::resetPixelValues(), Scheduler::run(), ModelCoreSpatial::runBiologicalModule(), ModelCore::runManagementModule(), ModelCoreSpatial::runManagementModule(), ModelCore::runMarketModule(), ModelCoreSpatial::runMarketModule(), Init::setInitLevel1(), Init::setInitLevel3(), ModelCore::sfd(), ModelCoreSpatial::sfd(), ModelCore::spd(), ModelCoreSpatial::spd(), ModelCoreSpatial::sumRegionalForData(), ModelCore::updateMapAreas(), ModelCoreSpatial::updateMapAreas(), and ModelCoreSpatial::updateOtherMapData().

4.42.4.12 QString messageStr [private]

Definition at line 184 of file ThreadManager.h.

4.42.4.13 QMutex mutex [private]

Definition at line 192 of file ThreadManager.h.

4.42.4.14 Ipopt::SmartPtr<Ipopt::TNLP> OPT

Market optimisation.

Definition at line 81 of file ThreadManager.h.

Referenced by ModelCore::runMarketModule(), and ModelCoreSpatial::runMarketModule().

4.42.4.15 volatile int pxQueryID [private]

Definition at line 190 of file ThreadManager.h.

4.42.4.16 volatile bool running [private]

Definition at line 186 of file ThreadManager.h.

4.42.4.17 Scheduler∗ SCD

the scheduler object (simulation-loops scheduler)

Definition at line 75 of file ThreadManager.h.

Referenced by ModelCoreSpatial::allocateHarvesting(), ModelCore::cacheSettings(), ModelCore::computeCumulativeData(), ModelCoreSpatial::computeCumulativeData(), ModelCore::computeInventary(), ModelCoreSpatial::computeInventary(), ModelData::getAllocableProductIdsFromDeathTimber(), ModelData::getBaseData(), Pixel::getMultiplier(), Pixel::getPathMortality(), Carbon::getStock(), ModelData::getTimedData(), Carbon::HWP_eol2energy(), Carbon::initialiseDeathBiomassStocks(), ModelCoreSpatial::initialiseDeathTimber(), Carbon::initialiseProductsStocks(), Output::print(), Layers::print(), Layers::printBinMap(), Output::printCarbonBalance(), Output::printDebugOutput(), Output::printDebugPixelValues(), Output::printForestData(), Output::printMaps(), Output::printOptLog(), Output::printProductData(), Carbon::registerDeathBiomass(), Carbon::registerHarvesting(), Carbon::registerProducts(), ModelCore::runBiologicalModule(), ModelCoreSpatial::runBiologicalModule(), ModelCore::runInitPeriod(), ModelCoreSpatial::runInitPeriod(), ModelCore::runManagementModule(), ModelCoreSpatial::runManagementModule(), ModelCore::runMarketModule(), ModelCoreSpatial::runMarketModule(), ModelCore::runSimulationYear(), ModelCoreSpatial::runSimulationYear(), Init::setInitLevel1(), Init::setInitLevel5(), ModelData::setTimedData(), ModelCoreSpatial::sumRegionalForData(), ModelCore::updateMapAreas(), and ModelCoreSpatial::updateMapAreas().

4.42.4.18 QString scenarioName [private]

Definition at line 189 of file ThreadManager.h.

4.42.4.19 ModelCoreSpatial∗ SCORE

Core of the model (spatial version)

Definition at line 78 of file ThreadManager.h.

Referenced by Scheduler::run(), and Init::setInitLevel3().

4.42.4.20 volatile bool stopped [private]

Definition at line 185 of file ThreadManager.h.

4.42.4.21 Sandbox∗ TEST

Various debugging code for development.

Definition at line 80 of file ThreadManager.h.

Referenced by Init::setInitLevel1().

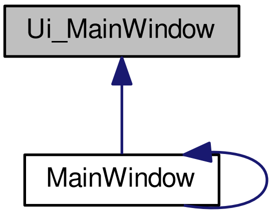
The documentation for this class was generated from the following files:

- /home/lobianco/git/ffsm_pp/src/ThreadManager.h
- /home/lobianco/git/ffsm_pp/src/ThreadManager.cpp

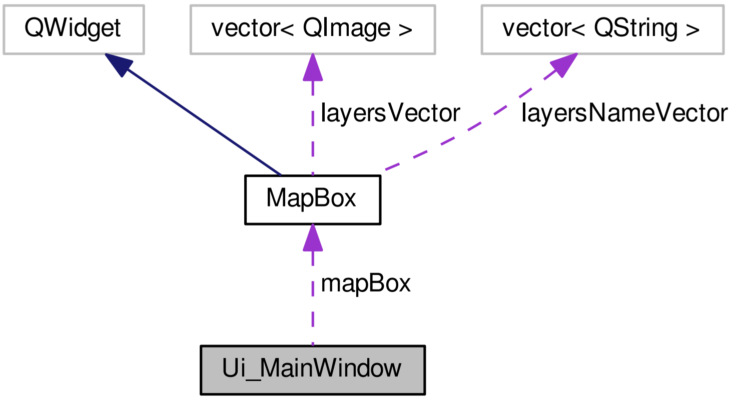
4.43 Ui_MainWindow Class Reference

`#include <ui_MainWindow.h>`

Inheritance diagram for Ui_MainWindow:



Collaboration diagram for Ui_MainWindow:



Public Member Functions

- void setupUi (QMainWindow ∗MainWindow)
- void retranslateUi (QMainWindow ∗MainWindow)

Public Attributes

- QAction ∗ actionLoadConfiguration
- QAction ∗ actionSaveLog
- QAction ∗ actionSaveLogAs

- QAction ∗ actionRun
- QAction ∗ actionPause
- QAction ∗ actionStop
- QAction ∗ actionAboutRegMAS
- QAction ∗ actionExit
- QAction ∗ actionHideDebugMsgs
- QAction ∗ actionRegMASDocumentation
- QAction ∗ actionFitMap
- QAction ∗ actionViewResults
- QWidget ∗ centralwidget
- QHBoxLayout ∗ hboxLayout
- QSplitter ∗ splitter
- QWidget ∗ layoutWidget
- QVBoxLayout ∗ vboxLayout
- QComboBox ∗ layerSelector
- QSpacerItem ∗ spacerItem
- MapBox ∗ mapBox
- QTabWidget ∗ tabWidget
- QWidget ∗ log_area
- QVBoxLayout ∗ verticalLayout
- QTextEdit ∗ logArea
- QPushButton ∗ viewResultsButton
- QWidget ∗ model_viewer
- QHBoxLayout ∗ hboxLayout1
- QTreeWidget ∗ statusView
- QWidget ∗ plot_info
- QGridLayout ∗ gridLayout
- QTextEdit ∗ pxInfoArea
- QMenuBar ∗ menubar
- QMenu ∗ menuView
- QMenu ∗ menuHelp
- QMenu ∗ menuAction
- QMenu ∗ menuFile
- QStatusBar ∗ statusbar
- QToolBar ∗ modelToolBar
- QToolBar ∗ fileToolBar

4.43.1 Detailed Description

Definition at line 38 of file ui_MainWindow.h.

4.43.2 Member Function Documentation

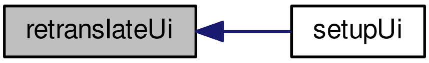
4.43.2.1 void retranslateUi (QMainWindow ∗ *MainWindow*) [inline]

Definition at line 292 of file ui_MainWindow.h.

Referenced by setupUi().

```
    {
        MainWindow->setWindowTitle(QApplication::translate("MainWindow", "FFSM - Forest Sector Simulator", 0));
        actionLoadConfiguration->setText(QApplication::translate("MainWindow", "&Load Configuration", 0));
        actionSaveLog->setText(QApplication::translate("MainWindow", "&Save log", 0));
        actionSaveLogAs->setText(QApplication::translate("MainWindow", "Save log &as..", 0));
        actionRun->setText(QApplication::translate("MainWindow", "&Run", 0));
        actionPause->setText(QApplication::translate("MainWindow", "&Pause / Resume", 0));
        actionStop->setText(QApplication::translate("MainWindow", "&Stop", 0));
        actionAboutRegMAS->setText(QApplication::translate("MainWindow", "&About RegMAS", 0));
        actionExit->setText(QApplication::translate("MainWindow", "&Exit", 0));
        actionHideDebugMsgs->setText(QApplication::translate("MainWindow", "Hide &debug messages", 0));
        actionRegMASDocumentation->setText(QApplication::translate("MainWindow", "RegMAS &documentation", 0));
        actionFitMap->setText(QApplication::translate("MainWindow", "&Fit map in Window", 0));
        actionViewResults->setText(QApplication::translate("MainWindow", "goToResults", 0));
#ifndef QT_NO_WHATSTHIS
        logArea->setWhatsThis(QApplication::translate("MainWindow", "<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:'Sans Serif'; font-size:9pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\">Run-time logs collecting area (can be saved)</p></body></html>", 0));
#endif // QT_NO_WHATSTHIS
#ifndef QT_NO_TOOLTIP
        viewResultsButton->setToolTip(QApplication::translate("MainWindow", "<html><head/><body><p>You will need a recent version of LibreOffice (or OpenOffice) installed on your system to view the results.</p><p>If you don't have it you can download it from <a href=\"http://www.libreoffice.org\"><span style=\" text-decoration: underline; color:#0000ff;\">http://www.libreoffice.org.</span></a></p><p/></body></html>", 0));
#endif // QT_NO_TOOLTIP
        viewResultsButton->setText(QApplication::translate("MainWindow", "Go to results", 0));
        tabWidget->setTabText(tabWidget->indexOf(log_area), QApplication::translate("MainWindow", "Log area", 0));
        QTreeWidgetItem *___qtreewidgetitem = statusView->headerItem();
        ___qtreewidgetitem->setText(0, QApplication::translate("MainWindow", "1", 0));
#ifndef QT_NO_WHATSTHIS
        statusView->setWhatsThis(QApplication::translate("MainWindow", "<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:'Sans Serif'; font-size:9pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\">Run-time viewer of important model status variables</p></body></html>", 0));
#endif // QT_NO_WHATSTHIS
        tabWidget->setTabText(tabWidget->indexOf(model_viewer), QApplication::translate("MainWindow", "Model viewer", 0));
        tabWidget->setTabText(tabWidget->indexOf(plot_info), QApplication::translate("MainWindow", "Plot info", 0));
        menuView->setTitle(QApplication::translate("MainWindow", "&View", 0));
        menuHelp->setTitle(QApplication::translate("MainWindow", "&Help", 0));
        menuAction->setTitle(QApplication::translate("MainWindow", "&Action", 0));
        menuFile->setTitle(QApplication::translate("MainWindow", "&File", 0));
    } // retranslateUi
```

Here is the caller graph for this function:

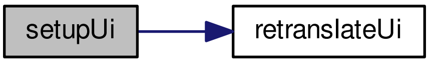


4.43.2.2 void setupUi (QMainWindow * *MainWindow*) [inline]

Definition at line 81 of file ui_MainWindow.h.

```
    {
        if (MainWindow->objectName().isEmpty())
            MainWindow->setObjectName(QStringLiteral("MainWindow"));
        MainWindow->setWindowModality(Qt::ApplicationModal);
        MainWindow->resize(667, 467);
        QSizePolicy sizePolicy(QSizePolicy::Fixed, QSizePolicy::Fixed);
        sizePolicy.setHorizontalStretch(1);
        sizePolicy.setVerticalStretch(1);
        sizePolicy.setHeightForWidth(MainWindow->sizePolicy().hasHeightForWidth());
        MainWindow->setSizePolicy(sizePolicy);
        QIcon icon;
        icon.addFile(QStringLiteral(":/imgs/icon.png"), QSize(), QIcon::Normal, QIcon::Off);
        MainWindow->setWindowIcon(icon);
        MainWindow->setIconSize(QSize(24, 24));
        actionLoadConfiguration = new QAction(MainWindow);
        actionLoadConfiguration->setObjectName(QStringLiteral("actionLoadConfiguration"));
        QIcon icon1;
        icon1.addFile(QStringLiteral(":/imgs/open.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionLoadConfiguration->setIcon(icon1);
        actionSaveLog = new QAction(MainWindow);
        actionSaveLog->setObjectName(QStringLiteral("actionSaveLog"));
        QIcon icon2;
        icon2.addFile(QStringLiteral(":/imgs/save.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionSaveLog->setIcon(icon2);
        actionSaveLogAs = new QAction(MainWindow);
        actionSaveLogAs->setObjectName(QStringLiteral("actionSaveLogAs"));
        QIcon icon3;
        icon3.addFile(QStringLiteral(":/imgs/saveas.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionSaveLogAs->setIcon(icon3);
        actionRun = new QAction(MainWindow);
        actionRun->setObjectName(QStringLiteral("actionRun"));
        QIcon icon4;
        icon4.addFile(QStringLiteral(":/imgs/play.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionRun->setIcon(icon4);
        actionPause = new QAction(MainWindow);
        actionPause->setObjectName(QStringLiteral("actionPause"));
        QIcon icon5;
        icon5.addFile(QStringLiteral(":/imgs/pause.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionPause->setIcon(icon5);
        actionStop = new QAction(MainWindow);
        actionStop->setObjectName(QStringLiteral("actionStop"));
        QIcon icon6;
        icon6.addFile(QStringLiteral(":/imgs/stop.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionStop->setIcon(icon6);
        actionAboutRegMAS = new QAction(MainWindow);
        actionAboutRegMAS->setObjectName(QStringLiteral("actionAboutRegMAS"));
        QIcon icon7;
        icon7.addFile(QStringLiteral(":/imgs/info.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionAboutRegMAS->setIcon(icon7);
        actionExit = new QAction(MainWindow);
        actionExit->setObjectName(QStringLiteral("actionExit"));
        QIcon icon8;
        icon8.addFile(QStringLiteral(":/imgs/exit.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionExit->setIcon(icon8);
        actionHideDebugMsgs = new QAction(MainWindow);
        actionHideDebugMsgs->setObjectName(QStringLiteral("actionHideDebugMsgs"));
        actionHideDebugMsgs->setCheckable(true);
        QIcon icon9;
        icon9.addFile(QStringLiteral(":/imgs/clear.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionHideDebugMsgs->setIcon(icon9);
        actionRegMASDocumentation = new QAction(MainWindow);
        actionRegMASDocumentation->setObjectName(QStringLiteral("actionRegMASDocumentation"));
        QIcon icon10;
        icon10.addFile(QStringLiteral(":/imgs/help.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionRegMASDocumentation->setIcon(icon10);
        actionFitMap = new QAction(MainWindow);
        actionFitMap->setObjectName(QStringLiteral("actionFitMap"));
        QIcon icon11;
        icon11.addFile(QStringLiteral(":/imgs/view-refresh.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionFitMap->setIcon(icon11);
        actionViewResults = new QAction(MainWindow);
        actionViewResults->setObjectName(QStringLiteral("actionViewResults"));
        centralwidget = new QWidget(MainWindow);
        centralwidget->setObjectName(QStringLiteral("centralwidget"));
        sizePolicy.setHeightForWidth(centralwidget->sizePolicy().hasHeightForWidth());
        centralwidget->setSizePolicy(sizePolicy);
        hboxLayout = new QHBoxLayout(centralwidget);
        hboxLayout->setObjectName(QStringLiteral("hboxLayout"));
        splitter = new QSplitter(centralwidget);
        splitter->setObjectName(QStringLiteral("splitter"));
        splitter->setOrientation(Qt::Horizontal);
        layoutWidget = new QWidget(splitter);
        layoutWidget->setObjectName(QStringLiteral("layoutWidget"));
        vboxLayout = new QVBoxLayout(layoutWidget);
        vboxLayout->setObjectName(QStringLiteral("vboxLayout"));
        vboxLayout->setContentsMargins(0, 0, 0, 0);
        layerSelector = new QComboBox(layoutWidget);
        layerSelector->setObjectName(QStringLiteral("layerSelector"));
        QSizePolicy sizePolicy1(QSizePolicy::Preferred, QSizePolicy::Fixed);
        sizePolicy1.setHorizontalStretch(1);
        sizePolicy1.setVerticalStretch(0);
        sizePolicy1.setHeightForWidth(layerSelector->sizePolicy().hasHeightForWidth());
        layerSelector->setSizePolicy(sizePolicy1);

        vboxLayout->addWidget(layerSelector);

        spacerItem = new QSpacerItem(200, 16, QSizePolicy::Minimum, QSizePolicy::Expanding);

        vboxLayout->addItem(spacerItem);

        mapBox = new MapBox(layoutWidget);
        mapBox->setObjectName(QStringLiteral("mapBox"));
        QSizePolicy sizePolicy2(QSizePolicy::Expanding, QSizePolicy::Expanding);
        sizePolicy2.setHorizontalStretch(2);
        sizePolicy2.setVerticalStretch(2);
        sizePolicy2.setHeightForWidth(mapBox->sizePolicy().hasHeightForWidth());
        mapBox->setSizePolicy(sizePolicy2);
        mapBox->setMinimumSize(QSize(300, 300));

        vboxLayout->addWidget(mapBox);

        splitter->addWidget(layoutWidget);
        tabWidget = new QTabWidget(splitter);
        tabWidget->setObjectName(QStringLiteral("tabWidget"));
        log_area = new QWidget();
        log_area->setObjectName(QStringLiteral("log_area"));
        verticalLayout = new QVBoxLayout(log_area);
        verticalLayout->setObjectName(QStringLiteral("verticalLayout"));
        logArea = new QTextEdit(log_area);
        logArea->setObjectName(QStringLiteral("logArea"));

        verticalLayout->addWidget(logArea);

        viewResultsButton = new QPushButton(log_area);
        viewResultsButton->setObjectName(QStringLiteral("viewResultsButton"));
        viewResultsButton->setLocale(QLocale(QLocale::English, QLocale::UnitedKingdom));

        verticalLayout->addWidget(viewResultsButton);

        tabWidget->addTab(log_area, QString());
        model_viewer = new QWidget();
        model_viewer->setObjectName(QStringLiteral("model_viewer"));
        hboxLayout1 = new QHBoxLayout(model_viewer);
        hboxLayout1->setObjectName(QStringLiteral("hboxLayout1"));
        statusView = new QTreeWidget(model_viewer);
        statusView->setObjectName(QStringLiteral("statusView"));

        hboxLayout1->addWidget(statusView);

        tabWidget->addTab(model_viewer, QString());
        plot_info = new QWidget();
        plot_info->setObjectName(QStringLiteral("plot_info"));
        gridLayout = new QGridLayout(plot_info);
        gridLayout->setObjectName(QStringLiteral("gridLayout"));
        pxInfoArea = new QTextEdit(plot_info);
        pxInfoArea->setObjectName(QStringLiteral("pxInfoArea"));
        pxInfoArea->setOverwriteMode(false);
        pxInfoArea->setTextInteractionFlags(Qt::TextSelectableByKeyboard|
Qt::TextSelectableByMouse);

        gridLayout->addWidget(pxInfoArea, 0, 0, 1, 1);

        tabWidget->addTab(plot_info, QString());
        splitter->addWidget(tabWidget);

        hboxLayout->addWidget(splitter);

        MainWindow->setCentralWidget(centralwidget);
        menubar = new QMenuBar(MainWindow);
        menubar->setObjectName(QStringLiteral("menubar"));
        menubar->setGeometry(QRect(0, 0, 667, 25));
        menuView = new QMenu(menubar);
        menuView->setObjectName(QStringLiteral("menuView"));
        menuHelp = new QMenu(menubar);
        menuHelp->setObjectName(QStringLiteral("menuHelp"));
        menuAction = new QMenu(menubar);
        menuAction->setObjectName(QStringLiteral("menuAction"));
        menuFile = new QMenu(menubar);
        menuFile->setObjectName(QStringLiteral("menuFile"));
        MainWindow->setMenuBar(menubar);
        statusbar = new QStatusBar(MainWindow);
        statusbar->setObjectName(QStringLiteral("statusbar"));
        MainWindow->setStatusBar(statusbar);
        modelToolBar = new QToolBar(MainWindow);
        modelToolBar->setObjectName(QStringLiteral("modelToolBar"));
        modelToolBar->setOrientation(Qt::Horizontal);
        MainWindow->addToolBar(Qt::TopToolBarArea, modelToolBar);
        fileToolBar = new QToolBar(MainWindow);
        fileToolBar->setObjectName(QStringLiteral("fileToolBar"));
        fileToolBar->setOrientation(Qt::Horizontal);
        MainWindow->addToolBar(Qt::TopToolBarArea, fileToolBar);

        menubar->addAction(menuFile->menuAction());
        menubar->addAction(menuAction->menuAction());
        menubar->addAction(menuView->menuAction());
        menubar->addAction(menuHelp->menuAction());
        menuView->addAction(actionHideDebugMsgs);
        menuView->addAction(actionFitMap);
        menuHelp->addAction(actionRegMASDocumentation);
        menuHelp->addAction(actionAboutRegMAS);
        menuAction->addAction(actionRun);
        menuAction->addAction(actionPause);
        menuAction->addAction(actionStop);
        menuFile->addAction(actionLoadConfiguration);
        menuFile->addAction(actionSaveLog);
        menuFile->addAction(actionSaveLogAs);
        modelToolBar->addAction(actionRun);
        modelToolBar->addAction(actionPause);
        modelToolBar->addAction(actionStop);
        fileToolBar->addAction(actionLoadConfiguration);
        fileToolBar->addAction(actionSaveLog);
        fileToolBar->addAction(actionExit);

        retranslateUi(MainWindow);

        tabWidget->setCurrentIndex(0);


        QMetaObject::connectSlotsByName(MainWindow);
    } // setupUi
```

Here is the call graph for this function:



4.43.3 Member Data Documentation

4.43.3.1 QAction∗ actionAboutRegMAS

Definition at line 47 of file ui_MainWindow.h.

4.43.3.2 QAction∗ actionExit

Definition at line 48 of file ui_MainWindow.h.

4.43.3.3 QAction∗ actionFitMap

Definition at line 51 of file ui_MainWindow.h.

4.43.3.4 QAction∗ actionHideDebugMsgs

Definition at line 49 of file ui_MainWindow.h.

4.43.3.5 QAction∗ actionLoadConfiguration

Definition at line 41 of file ui_MainWindow.h.

4.43.3.6 QAction∗ actionPause

Definition at line 45 of file ui_MainWindow.h.

4.43.3.7 QAction∗ actionRegMASDocumentation

Definition at line 50 of file ui_MainWindow.h.

4.43.3.8 QAction∗ actionRun

Definition at line 44 of file ui_MainWindow.h.

4.43.3.9 QAction∗ actionSaveLog

Definition at line 42 of file ui_MainWindow.h.

4.43.3.10 QAction∗ actionSaveLogAs

Definition at line 43 of file ui_MainWindow.h.

4.43.3.11 QAction∗ actionStop

Definition at line 46 of file ui_MainWindow.h.

4.43.3.12 QAction∗ actionViewResults

Definition at line 52 of file ui_MainWindow.h.

4.43.3.13 QWidget∗ centralwidget

Definition at line 53 of file ui_MainWindow.h.

4.43.3.14 QToolBar∗ fileToolBar

Definition at line 79 of file ui_MainWindow.h.

4.43.3.15 QGridLayout∗ gridLayout

Definition at line 70 of file ui_MainWindow.h.

4.43.3.16 QHBoxLayout∗ hboxLayout

Definition at line 54 of file ui_MainWindow.h.

4.43.3.17 QHBoxLayout∗ hboxLayout1

Definition at line 67 of file ui_MainWindow.h.

4.43.3.18 QComboBox∗ layerSelector

Definition at line 58 of file ui_MainWindow.h.

4.43.3.19 QWidget∗ layoutWidget

Definition at line 56 of file ui_MainWindow.h.

4.43.3.20 QWidget∗ log_area

Definition at line 62 of file ui_MainWindow.h.

4.43.3.21 QTextEdit∗ logArea

Definition at line 64 of file ui_MainWindow.h.

4.43.3.22 MapBox∗ mapBox

Definition at line 60 of file ui_MainWindow.h.

4.43.3.23 QMenu∗ menuAction

Definition at line 75 of file ui_MainWindow.h.

4.43.3.24 QMenuBar∗ menubar

Definition at line 72 of file ui_MainWindow.h.

4.43.3.25 QMenu∗ menuFile

Definition at line 76 of file ui_MainWindow.h.

4.43.3.26 QMenu∗ menuHelp

Definition at line 74 of file ui_MainWindow.h.

4.43.3.27 QMenu∗ menuView

Definition at line 73 of file ui_MainWindow.h.

4.43.3.28 QWidget∗ model_viewer

Definition at line 66 of file ui_MainWindow.h.

4.43.3.29 QToolBar∗ modelToolBar

Definition at line 78 of file ui_MainWindow.h.

4.43.3.30 QWidget∗ plot_info

Definition at line 69 of file ui_MainWindow.h.

4.43.3.31 QTextEdit∗ pxInfoArea

Definition at line 71 of file ui_MainWindow.h.

4.43.3.32 QSpacerItem∗ spacerItem

Definition at line 59 of file ui_MainWindow.h.

4.43.3.33 QSplitter∗ splitter

Definition at line 55 of file ui_MainWindow.h.

4.43.3.34 QStatusBar∗ statusbar

Definition at line 77 of file ui_MainWindow.h.

4.43.3.35 QTreeWidget∗ statusView

Definition at line 68 of file ui_MainWindow.h.

4.43.3.36 QTabWidget∗ tabWidget

Definition at line 61 of file ui_MainWindow.h.

4.43.3.37 QVBoxLayout∗ vboxLayout

Definition at line 57 of file ui_MainWindow.h.

4.43.3.38 QVBoxLayout∗ verticalLayout

Definition at line 63 of file ui_MainWindow.h.

4.43.3.39 QPushButton∗ viewResultsButton

Definition at line 65 of file ui_MainWindow.h.

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

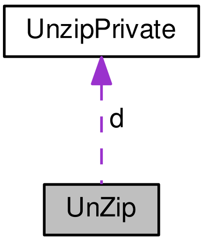
- /home/lobianco/git/ffsm_pp/src/ui_MainWindow.h

4.44 UnZip Class Reference

PKZip 2.0 file decompression. Compatibility with later versions is not ensured as they may use unsupported compression algorithms. Versions after 2.7 may have an incompatible header format and thus be completely incompatible.

`#include <unzip.h>`

Collaboration diagram for UnZip:



Classes

- struct ZipEntry

Public Types

- enum ErrorCode {
  Ok, ZlibInit, ZlibError, OpenFailed,
  PartiallyCorrupted, Corrupted, WrongPassword, NoOpenArchive,
  FileNotFound, ReadFailed, WriteFailed, SeekFailed,
  CreateDirFailed, InvalidDevice, InvalidArchive, HeaderConsistencyError,
  Skip, SkipAll }
- enum ExtractionOption { ExtractPaths = 0x0001, SkipPaths = 0x0002 }
- enum CompressionMethod { NoCompression, Deflated, UnknownCompression }
- enum FileType { File, Directory }

Public Member Functions

- UnZip ()
- virtual ∼UnZip ()
- bool isOpen () const
- ErrorCode openArchive (const QString &filename)
- ErrorCode openArchive (QIODevice ∗device)
- void closeArchive ()
- QString archiveComment () const
- QString formatError (UnZip::ErrorCode c) const
- bool contains (const QString &file) const

- QStringList fileList () const
- QList< ZipEntry > entryList () const
- ErrorCode extractAll (const QString &dirname, ExtractionOptions options=ExtractPaths)
- ErrorCode extractAll (const QDir &dir, ExtractionOptions options=ExtractPaths)
- ErrorCode extractFile (const QString &filename, const QString &dirname, ExtractionOptions options=ExtractPaths)
- ErrorCode extractFile (const QString &filename, const QDir &dir, ExtractionOptions options=ExtractPaths)
- ErrorCode extractFile (const QString &filename, QIODevice ∗device, ExtractionOptions options=ExtractPaths)
- ErrorCode extractFiles (const QStringList &filenames, const QString &dirname, ExtractionOptions options=ExtractPaths)
- ErrorCode extractFiles (const QStringList &filenames, const QDir &dir, ExtractionOptions options=ExtractPaths)
- void setPassword (const QString &pwd)

Private Attributes

- UnzipPrivate ∗ d

4.44.1 Detailed Description

PKZip 2.0 file decompression. Compatibility with later versions is not ensured as they may use unsupported compression algorithms. Versions after 2.7 may have an incompatible header format and thus be completely incompatible.

Definition at line 45 of file unzip.h.

4.44.2 Member Enumeration Documentation

4.44.2.1 enum CompressionMethod

Enumerator

NoCompression

Deflated

UnknownCompression

Definition at line 79 of file unzip.h.

```
00080   {
00081     NoCompression, Deflated, UnknownCompression
00082   };
```

4.44.2.2 enum ErrorCode

The result of a decompression operation. UnZip::Ok No error occurred. UnZip::ZlibInit Failed to init or load the zlib library. UnZip::ZlibError The zlib library returned some error. UnZip::OpenFailed Unable to create or open a device. UnZip::PartiallyCorrupted Corrupted zip archive - some files could be extracted. UnZip::Corrupted Corrupted or invalid zip archive. UnZip::WrongPassword Unable to decrypt a password protected file. UnZip::NoOpenArchive No archive has been opened yet. UnZip::FileNotFound Unable to find the requested file in the archive. UnZip::ReadFailed Reading of a file failed. UnZip::WriteFailed Writing of a file failed. UnZip::SeekFailed Seek failed. UnZip::CreateDirFailed Could not create a directory. UnZip::InvalidDevice A null device has been passed as parameter. UnZip::InvalidArchive This is not a valid (or supported) ZIP archive. UnZip::HeaderConsistencyError Local header record info does not match with the central directory record info. The archive may be corrupted.

UnZip::Skip Internal use only. UnZip::SkipAll Internal use only.

Enumerator

- ***Ok***
- ***ZlibInit***
- ***ZlibError***
- ***OpenFailed***
- ***PartiallyCorrupted***
- ***Corrupted***
- ***WrongPassword***
- ***NoOpenArchive***
- ***FileNotFound***
- ***ReadFailed***
- ***WriteFailed***
- ***SeekFailed***
- ***CreateDirFailed***
- ***InvalidDevice***
- ***InvalidArchive***
- ***HeaderConsistencyError***
- ***Skip***
- ***SkipAll***

Definition at line 48 of file unzip.h.

```
00049   {
00050     Ok,
00051     ZlibInit,
00052     ZlibError,
00053     OpenFailed,
00054     PartiallyCorrupted,
00055     Corrupted,
00056     WrongPassword,
00057     NoOpenArchive,
00058     FileNotFound,
00059     ReadFailed,
00060     WriteFailed,
00061     SeekFailed,
00062     CreateDirFailed,
00063     InvalidDevice,
00064     InvalidArchive,
00065     HeaderConsistencyError,
00066 
00067     Skip, SkipAll // internal use only
00068   };
```

4.44.2.3 enum ExtractionOption

Enumerator

ExtractPaths	Extracts paths (default)
SkipPaths	Ignores paths and extracts all the files to the same directory.

Definition at line 70 of file unzip.h.

```
  {
    //! Extracts paths (default)
    ExtractPaths = 0x0001,
    //! Ignores paths and extracts all the files to the same directory
    SkipPaths = 0x0002
  };
```

4.44.2.4 enum FileType

Enumerator

File

Directory

Definition at line 84 of file unzip.h.

```
  {
    File, Directory
  };
```

4.44.3 Constructor & Destructor Documentation

4.44.3.1 UnZip ( )

Creates a new Zip file decompressor.

Definition at line 165 of file unzip.cpp.

```
{
  d = new UnzipPrivate;
}
```

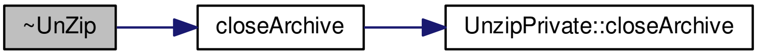
4.44.3.2 ~UnZip ( ) [virtual]

Closes any open archive and releases used resources.

Definition at line 173 of file unzip.cpp.

```
{
  closeArchive();
  delete d;
}
```

Here is the call graph for this function:



4.44.4 Member Function Documentation

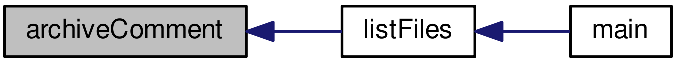
4.44.4.1 QString archiveComment ( ) const

Definition at line 231 of file unzip.cpp.

Referenced by listFiles().

```
00232 {
00233   if (d->device == 0)
00234     return QString();
00235   return d->comment;
00236 }
```

Here is the caller graph for this function:



4.44.4.2 void closeArchive ( )

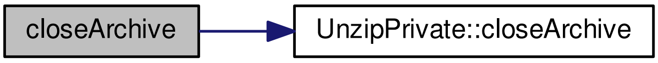
Closes the archive and releases all the used resources (like cached passwords).

Definition at line 226 of file unzip.cpp.

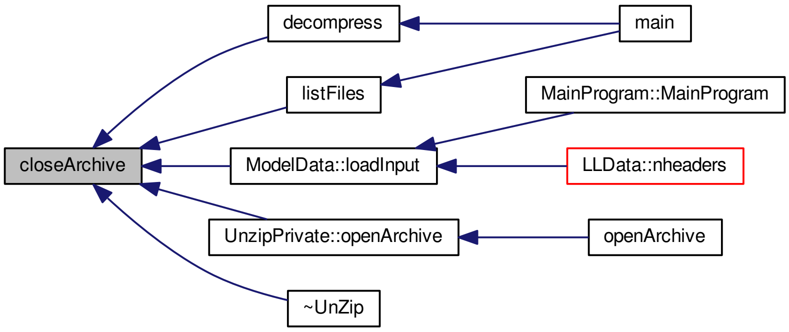
Referenced by decompress(), listFiles(), ModelData::loadInput(), UnzipPrivate::openArchive(), and ∼UnZip().

```
00227 {
00228   d->closeArchive();
00229 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.44.4.3 bool contains (const QString & *file*) const

Returns true if the archive contains a file with the given path and name.

Definition at line 270 of file unzip.cpp.

```
{
  if (d->headers == 0)
    return false;

  return d->headers->contains(file);
}
```

4.44.4.4 QList< UnZip::ZipEntry > entryList ( ) const

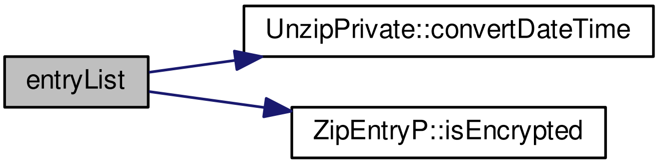
Returns information for each (correctly parsed) entry of this archive.

Definition at line 289 of file unzip.cpp.

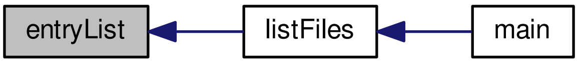
Referenced by listFiles().

```
{
  QList<UnZip::ZipEntry> list;

  if (d->headers != 0)
  {
    for (QMap<QString,ZipEntryP*>::ConstIterator it = d->headers->constBegin(); it != d->headers->constEnd(); ++it)
    {
      const ZipEntryP* entry = it.value();
      Q_ASSERT(entry != 0);

      ZipEntry z;

      z.filename = it.key();
      if (!entry->comment.isEmpty())
        z.comment = entry->comment;
      z.compressedSize = entry->szComp;
      z.uncompressedSize = entry->szUncomp;
      z.crc32 = entry->crc;
      z.lastModified = d->convertDateTime(entry->modDate, entry->modTime);

      z.compression = entry->compMethod == 0 ? NoCompression : entry->compMethod == 8 ? Deflated : UnknownCompression;
      z.type = z.filename.endsWith("/") ? Directory : File;

      z.encrypted = entry->isEncrypted();

      list.append(z);
    }
  }

  return list;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.44.4.5 UnZip::ErrorCode extractAll (const QString & *dirname*, ExtractionOptions *options* = ExtractPaths)

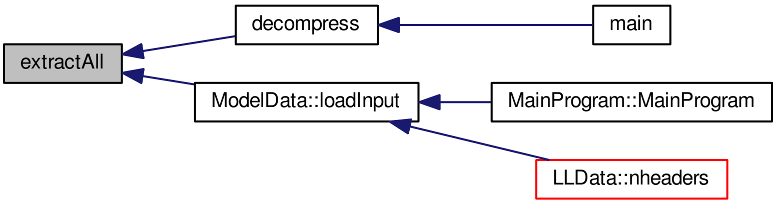
Extracts the whole archive to a directory.

Definition at line 325 of file unzip.cpp.

Referenced by decompress(), and ModelData::loadInput().

```
{
  return extractAll(QDir(dirname), options);
}
```

Here is the caller graph for this function:



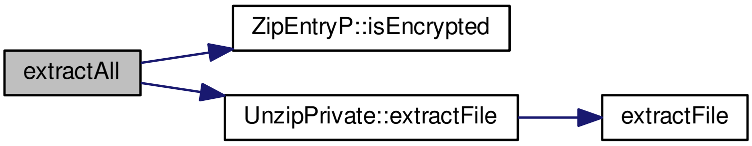
4.44.4.6 UnZip::ErrorCode extractAll (const QDir & *dir*, ExtractionOptions *options* = ExtractPaths)

Extracts the whole archive to a directory.

Definition at line 333 of file unzip.cpp.

```
{
  // this should only happen if we didn't call openArchive() yet
  if (d->device == 0)
    return NoOpenArchive;

  if (d->headers == 0)
    return Ok;

  bool end = false;
  for (QMap<QString,ZipEntryP*>::Iterator itr = d->headers->begin(); itr !=
d->headers->end(); ++itr)
  {
    ZipEntryP* entry = itr.value();
    Q_ASSERT(entry != 0);

    if ((entry->isEncrypted()) && d->skipAllEncrypted)
      continue;

    switch (d->extractFile(itr.key(), *entry, dir, options))
    {
    case Corrupted:
      qDebug() << "Removing corrupted entry" << itr.key();
      d->headers->erase(itr++);
      if (itr == d->headers->end())
        end = true;
      break;
    case CreateDirFailed:
      break;
    case Skip:
      break;
    case SkipAll:
      d->skipAllEncrypted = true;
      break;
    default:
      ;
    }

    if (end)
      break;
  }

  return Ok;
}
```

Here is the call graph for this function:



4.44.4.7 UnZip::ErrorCode extractFile (const QString & *filename*, const QString & *dirname*, ExtractionOptions *options* = ExtractPaths)

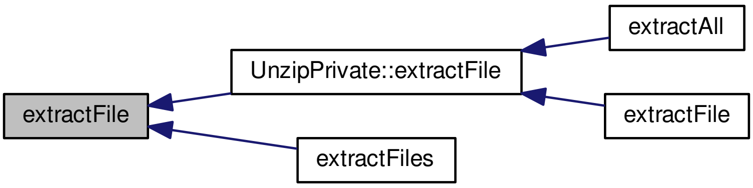
Extracts a single file to a directory.

Definition at line 380 of file unzip.cpp.

Referenced by UnzipPrivate::extractFile(), and extractFiles().

```
{
  return extractFile(filename, QDir(dirname), options);
}
```

Here is the caller graph for this function:



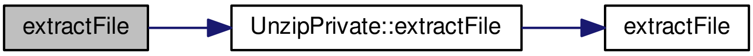
4.44.4.8 UnZip::ErrorCode extractFile (const QString & *filename*, const QDir & *dir*, ExtractionOptions *options* = ExtractPaths)

Extracts a single file to a directory.

Definition at line 388 of file unzip.cpp.

```
{
  QMap<QString,ZipEntryP*>::Iterator itr = d->headers->find(filename);
  if (itr != d->headers->end())
  {
    ZipEntryP* entry = itr.value();
    Q_ASSERT(entry != 0);
    return d->extractFile(itr.key(), *entry, dir, options);
  }

  return FileNotFound;
}
```

Here is the call graph for this function:



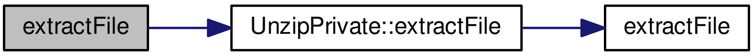
4.44.4.9 UnZip::ErrorCode extractFile (const QString & *filename*, QIODevice ∗ *dev*, ExtractionOptions *options* = ExtractPaths)

Extracts a single file to a directory.

Definition at line 404 of file unzip.cpp.

```
00405 {
00406   if (dev == 0)
00407     return InvalidDevice;
00408
00409   QMap<QString,ZipEntryP*>::Iterator itr = d->headers->find(filename);
00410   if (itr != d->headers->end()) {
00411     ZipEntryP* entry = itr.value();
00412     Q_ASSERT(entry != 0);
00413     return d->extractFile(itr.key(), *entry, dev, options);
00414   }
00415
00416   return FileNotFound;
00417 }
```

Here is the call graph for this function:



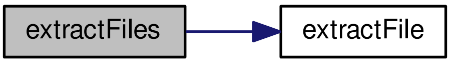
4.44.4.10 UnZip::ErrorCode extractFiles (const QStringList & *filenames*, const QString & *dirname*, ExtractionOptions *options* = ExtractPaths)

Extracts a list of files. Stops extraction at the first error (but continues if a file does not exist in the archive).

Definition at line 423 of file unzip.cpp.

```
00424 {
00425   QDir dir(dirname);
00426   ErrorCode ec;
00427
00428   for (QStringList::ConstIterator itr = filenames.constBegin(); itr != filenames.constEnd(); ++itr)
00429   {
00430     ec = extractFile(*itr, dir, options);
00431     if (ec == FileNotFound)
00432       continue;
00433     if (ec != Ok)
00434       return ec;
00435   }
00436
00437   return Ok;
00438 }
```

Here is the call graph for this function:



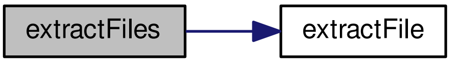
4.44.4.11 UnZip::ErrorCode extractFiles (const QStringList & *filenames*, const QDir & *dir*, ExtractionOptions *options* = ExtractPaths)

Extracts a list of files. Stops extraction at the first error (but continues if a file does not exist in the archive).

Definition at line 444 of file unzip.cpp.

```
{
  ErrorCode ec;

  for (QStringList::ConstIterator itr = filenames.constBegin(); itr != filenames.constEnd(); ++itr)
  {
    ec = extractFile(*itr, dir, options);
    if (ec == FileNotFound)
      continue;
    if (ec != Ok)
      return ec;
  }

  return Ok;
}
```

Here is the call graph for this function:



4.44.4.12 QStringList fileList ( ) const

Returns complete paths of files and directories in this archive.

Definition at line 281 of file unzip.cpp.

```
{
  return d->headers == 0 ? QStringList() : d->headers->keys();
}
```

4.44.4.13 QString formatError (UnZip::ErrorCode *c*) const

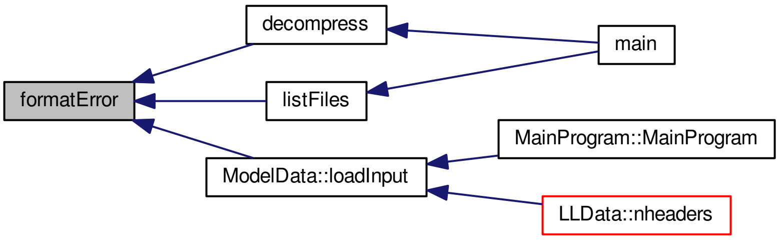
Returns a locale translated error string for a given error code.

Definition at line 241 of file unzip.cpp.

Referenced by decompress(), listFiles(), and ModelData::loadInput().

```
00242 {
00243   switch (c)
00244   {
00245   case Ok: return QCoreApplication::translate("UnZip", "ZIP operation completed successfully."); break;
00246   case ZlibInit: return QCoreApplication::translate("UnZip", "Failed to initialize or load zlib
      library."); break;
00247   case ZlibError: return QCoreApplication::translate("UnZip", "zlib library error."); break;
00248   case OpenFailed: return QCoreApplication::translate("UnZip", "Unable to create or open file.");
      break;
00249   case PartiallyCorrupted: return QCoreApplication::translate("UnZip", "Partially
      corrupted archive. Some files might be extracted."); break;
00250   case Corrupted: return QCoreApplication::translate("UnZip", "Corrupted archive."); break;
00251   case WrongPassword: return QCoreApplication::translate("UnZip", "Wrong password."); break;
00252   case NoOpenArchive: return QCoreApplication::translate("UnZip", "No archive has been created
      yet."); break;
00253   case FileNotFound: return QCoreApplication::translate("UnZip", "File or directory does not
      exist."); break;
00254   case ReadFailed: return QCoreApplication::translate("UnZip", "File read error."); break;
00255   case WriteFailed: return QCoreApplication::translate("UnZip", "File write error."); break;
00256   case SeekFailed: return QCoreApplication::translate("UnZip", "File seek error."); break;
00257   case CreateDirFailed: return QCoreApplication::translate("UnZip", "Unable to create a
      directory."); break;
00258   case InvalidDevice: return QCoreApplication::translate("UnZip", "Invalid device."); break;
00259   case InvalidArchive: return QCoreApplication::translate("UnZip", "Invalid or incompatible
      zip archive."); break;
00260   case HeaderConsistencyError: return QCoreApplication::translate("UnZip", "
     Inconsistent headers. Archive might be corrupted."); break;
00261   default: ;
00262   }
00263
00264   return QCoreApplication::translate("UnZip", "Unknown error.");
00265 }
```

Here is the caller graph for this function:



4.44.4.14 bool isOpen ( ) const

Returns true if there is an open archive.

Definition at line 182 of file unzip.cpp.

```
00183 {
00184   return d->device != 0;
00185 }
```

4.44.4.15 UnZip::ErrorCode openArchive (const QString & *filename*)

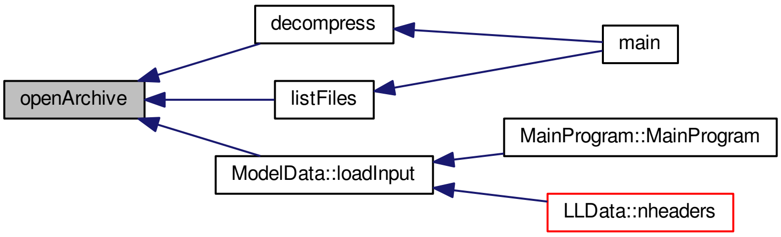
Opens a zip archive and reads the files list. Closes any previously opened archive.

Definition at line 190 of file unzip.cpp.

Referenced by decompress(), listFiles(), and ModelData::loadInput().

```
{
  QFile* file = new QFile(filename);

  if (!file->exists()) {
    delete file;
    return UnZip::FileNotFound;
  }

  if (!file->open(QIODevice::ReadOnly)) {
    delete file;
    return UnZip::OpenFailed;
  }

  return openArchive(file);
}
```

Here is the caller graph for this function:



4.44.4.16 UnZip::ErrorCode openArchive (QIODevice ∗ *device*)

Opens a zip archive and reads the entries list. Closes any previously opened archive.

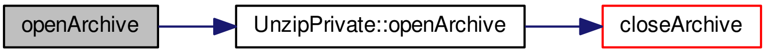
Warning

> The class takes ownership of the device so don't delete it!

Definition at line 212 of file unzip.cpp.

```
{
  if (device == 0)
  {
    qDebug() << "Invalid device.";
    return UnZip::InvalidDevice;
  }

  return d->openArchive(device);
}
```

Here is the call graph for this function:



4.44.4.17 void setPassword (const QString & *pwd*)

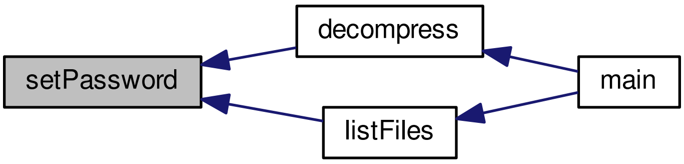
Remove/replace this method to add your own password retrieval routine.

Definition at line 463 of file unzip.cpp.

Referenced by decompress(), and listFiles().

```
00464 {
00465   d->password = pwd;
00466 }
```

Here is the caller graph for this function:



4.44.5 Member Data Documentation

4.44.5.1 UnzipPrivate∗ d [private]

Definition at line 139 of file unzip.h.

Referenced by archiveComment(), closeArchive(), contains(), UnzipPrivate::createDirectory(), entryList(), extractAll(), extractFile(), fileList(), isOpen(), openArchive(), setPassword(), UnZip(), and ∼UnZip().

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

- /home/lobianco/git/ffsm_pp/src/unzip.h
- /home/lobianco/git/ffsm_pp/src/unzip.cpp

4.45 UnzipPrivate Class Reference

`#include <unzip_p.h>`

Public Member Functions

- UnzipPrivate ()
- UnZip::ErrorCode openArchive (QIODevice ∗device)
- UnZip::ErrorCode seekToCentralDirectory ()
- UnZip::ErrorCode parseCentralDirectoryRecord ()
- UnZip::ErrorCode parseLocalHeaderRecord (const QString &path, ZipEntryP &entry)
- void closeArchive ()
- UnZip::ErrorCode extractFile (const QString &path, ZipEntryP &entry, const QDir &dir, UnZip::Extraction←Options options)
- UnZip::ErrorCode extractFile (const QString &path, ZipEntryP &entry, QIODevice ∗device, UnZip::←ExtractionOptions options)
- UnZip::ErrorCode testPassword (quint32 ∗keys, const QString &file, const ZipEntryP &header)
- bool testKeys (const ZipEntryP &header, quint32 ∗keys)
- bool createDirectory (const QString &path)
- void decryptBytes (quint32 ∗keys, char ∗buffer, qint64 read)
- quint32 getULong (const unsigned char ∗data, quint32 offset) const
- quint64 getULLong (const unsigned char ∗data, quint32 offset) const
- quint16 getUShort (const unsigned char ∗data, quint32 offset) const
- int decryptByte (quint32 key2) const
- void updateKeys (quint32 ∗keys, int c) const
- void initKeys (const QString &pwd, quint32 ∗keys) const
- QDateTime convertDateTime (const unsigned char date[2], const unsigned char time[2]) const

Public Attributes

- QString password
- bool skipAllEncrypted
- QMap< QString, ZipEntryP ∗ > ∗ headers
- QIODevice ∗ device
- char buffer1 [UNZIP_READ_BUFFER]
- char buffer2 [UNZIP_READ_BUFFER]
- unsigned char ∗ uBuffer
- const quint32 ∗ crcTable
- quint32 cdOffset
- quint32 eocdOffset
- quint16 cdEntryCount
- quint16 unsupportedEntryCount
- QString comment

4.45.1 Detailed Description

Definition at line 51 of file unzip_p.h.

4.45.2 Constructor & Destructor Documentation

4.45.2.1 UnzipPrivate ( )

Definition at line 485 of file unzip.cpp.

```
{
  skipAllEncrypted = false;
  headers = 0;
  device = 0;

  uBuffer = (unsigned char*) buffer1;
  crcTable = (quint32*) get_crc_table();

  cdOffset = eocdOffset = 0;
  cdEntryCount = 0;
  unsupportedEntryCount = 0;
}
```

4.45.3 Member Function Documentation

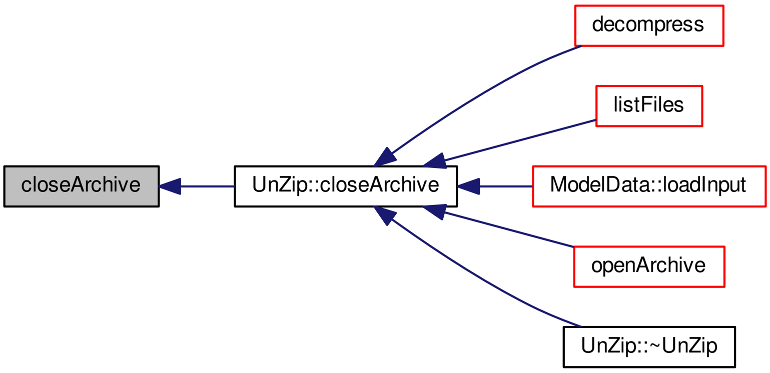
4.45.3.1 void closeArchive ( )

Definition at line 948 of file unzip.cpp.

Referenced by UnZip::closeArchive().

```
{
  if (device == 0)
    return;

  skipAllEncrypted = false;

  if (headers != 0)
  {
    qDeleteAll(*headers);
    delete headers;
    headers = 0;
  }

  delete device; device = 0;

  cdOffset = eocdOffset = 0;
  cdEntryCount = 0;
  unsupportedEntryCount = 0;

  comment.clear();
}
```

Here is the caller graph for this function:



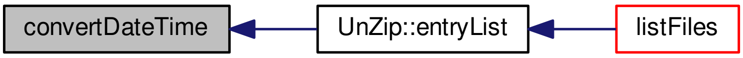
4.45.3.2 QDateTime convertDateTime (const unsigned char *date[2]*, const unsigned char *time[2]*) const [inline]

Definition at line 1345 of file unzip.cpp.

Referenced by UnZip::entryList().

```
{
  QDateTime dt;

  // Usual PKZip low-byte to high-byte order

  // Date: 7 bits = years from 1980, 4 bits = month, 5 bits = day
  quint16 year = (date[1] >> 1) & 127;
  quint16 month = ((date[1] << 3) & 14) | ((date[0] >> 5) & 7);
  quint16 day = date[0] & 31;

  // Time: 5 bits hour, 6 bits minutes, 5 bits seconds with a 2sec precision
  quint16 hour = (time[1] >> 3) & 31;
  quint16 minutes = ((time[1] << 3) & 56) | ((time[0] >> 5) & 7);
  quint16 seconds = (time[0] & 31) * 2;

  dt.setDate(QDate(1980 + year, month, day));
  dt.setTime(QTime(hour, minutes, seconds));
  return dt;
}
```

Here is the caller graph for this function:



4.45.3.3 bool createDirectory (const QString & *path*)

Definition at line 1195 of file unzip.cpp.

```
{
  QDir d(path);
  if (!d.exists())
  {
    int sep = path.lastIndexOf("/");
    if (sep <= 0) return true;

    if (!createDirectory(path.left(sep)))
      return false;

    if (!d.mkdir(path))
    {
      qDebug() << QString("Unable to create directory: %1").arg(path);
      return false;
    }
  }

  return true;
}
```

4.45.3.4 int decryptByte (quint32 *key2*) const [inline]

Definition at line 1257 of file unzip.cpp.

```
{
  quint16 temp = ((quint16)(key2) & 0xffff) | 2;
  return (int)(((temp * (temp ^ 1)) >> 8) & 0xff);
}
```

4.45.3.5 void decryptBytes (quint32 * *keys*, char * *buffer*, qint64 *read*) [inline]

Definition at line 1336 of file unzip.cpp.

```
{
  for (int i=0; i<(int)read; ++i)
    updateKeys(keys, buffer[i] ^= decryptByte(keys[2]));
}
```

4.45.3.6 UnZip::ErrorCode extractFile (const QString & *path*, ZipEntryP & *entry*, const QDir & *dir*, UnZip::ExtractionOptions *options*)

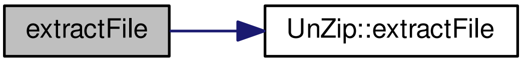
Todo Set creation/last_modified date/time

Definition at line 972 of file unzip.cpp.

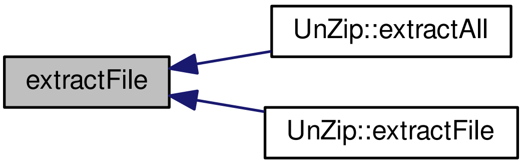
Referenced by UnZip::extractAll(), and UnZip::extractFile().

```
{
  QString name(path);
  QString dirname;
  QString directory;

  int pos = name.lastIndexOf('/');

  // This entry is for a directory
  if (pos == name.length() - 1)
  {
    if (options.testFlag(UnZip::SkipPaths))
      return UnZip::Ok;

    directory = QString("%1/%2").arg(dir.absolutePath()).arg(QDir::cleanPath(name));
    if (!createDirectory(directory))
    {
      qDebug() << QString("Unable to create directory: %1").arg(directory);
      return UnZip::CreateDirFailed;
    }

    return UnZip::Ok;
  }

  // Extract path from entry
  if (pos > 0)
  {
    // get directory part
    dirname = name.left(pos);
    if (options.testFlag(UnZip::SkipPaths))
    {
      directory = dir.absolutePath();
    }
    else
    {
      directory = QString("%1/%2").arg(dir.absolutePath()).arg(QDir::cleanPath(dirname));
      if (!createDirectory(directory))
      {
        qDebug() << QString("Unable to create directory: %1").arg(directory);
        return UnZip::CreateDirFailed;
      }
    }
    name = name.right(name.length() - pos - 1);
  } else directory = dir.absolutePath();

  name = QString("%1/%2").arg(directory).arg(name);

  QFile outFile(name);

  if (!outFile.open(QIODevice::WriteOnly))
  {
    qDebug() << QString("Unable to open %1 for writing").arg(name);
    return UnZip::OpenFailed;
  }

  //! \todo Set creation/last_modified date/time

  UnZip::ErrorCode ec = extractFile(path, entry, &outFile, options);

  outFile.close();

  if (ec != UnZip::Ok)
  {
    if (!outFile.remove())
      qDebug() << QString("Unable to remove corrupted file: %1").arg(name);
  }

  return ec;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.45.3.7 UnZip::ErrorCode extractFile (const QString & *path*, ZipEntryP & *entry*, QIODevice ∗ *device*, UnZip::ExtractionOptions *options*)

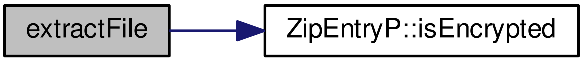
Encryption header size

Definition at line 1043 of file unzip.cpp.

```
{
  Q_UNUSED(options);
  Q_ASSERT(dev != 0);

  if (!entry.lhEntryChecked)
  {
    UnZip::ErrorCode ec = parseLocalHeaderRecord(path, entry);
    entry.lhEntryChecked = true;

    if (ec != UnZip::Ok)
      return ec;
  }

  if (!device->seek(entry.dataOffset))
    return UnZip::SeekFailed;

  // Encryption keys
  quint32 keys[3];

  if (entry.isEncrypted())
  {
    UnZip::ErrorCode e = testPassword(keys, path, entry);
    if (e != UnZip::Ok)
    {
      qDebug() << QString("Unable to decrypt %1").arg(path);
      return e;
    }//! Encryption header size
    entry.szComp -= UNZIP_LOCAL_ENC_HEADER_SIZE; // remove encryption header size
  }

  if (entry.szComp == 0)
  {
    if (entry.crc != 0)
      return UnZip::Corrupted;

    return UnZip::Ok;
  }

  uInt rep = entry.szComp / UNZIP_READ_BUFFER;
  uInt rem = entry.szComp % UNZIP_READ_BUFFER;
  uInt cur = 0;

  // extract data
  qint64 read;
  quint64 tot = 0;

  quint32 myCRC = crc32(0L, Z_NULL, 0);

  if (entry.compMethod == 0)
  {
    while ( (read = device->read(buffer1, cur < rep ? UNZIP_READ_BUFFER : rem)) > 0 )
    {
      if (entry.isEncrypted())
        decryptBytes(keys, buffer1, read);

      myCRC = crc32(myCRC, uBuffer, read);

      if (dev->write(buffer1, read) != read)
        return UnZip::WriteFailed;

      cur++;
      tot += read;

      if (tot == entry.szComp)
        break;
    }

    if (read < 0)
      return UnZip::ReadFailed;
  }
  else if (entry.compMethod == 8)
  {
    /* Allocate inflate state */
    z_stream zstr;
    zstr.zalloc = Z_NULL;
    zstr.zfree = Z_NULL;
    zstr.opaque = Z_NULL;
    zstr.next_in = Z_NULL;
    zstr.avail_in = 0;

    int zret;

    // Use inflateInit2 with negative windowBits to get raw decompression
    if ( (zret = inflateInit2_(&zstr, -MAX_WBITS, ZLIB_VERSION, sizeof(z_stream))) != Z_OK )
      return UnZip::ZlibError;

    int szDecomp;

    // Decompress until deflate stream ends or end of file
    do
    {
      read = device->read(buffer1, cur < rep ? UNZIP_READ_BUFFER : rem);
      if (read == 0)
        break;
      if (read < 0)
      {
        (void)inflateEnd(&zstr);
        return UnZip::ReadFailed;
      }

      if (entry.isEncrypted())
```

```
        decryptBytes(keys, buffer1, read);

      cur++;
      tot += read;

      zstr.avail_in = (uInt) read;
      zstr.next_in = (Bytef*) buffer1;


      // Run inflate() on input until output buffer not full
      do {
        zstr.avail_out = UNZIP_READ_BUFFER;
        zstr.next_out = (Bytef*) buffer2;;

        zret = inflate(&zstr, Z_NO_FLUSH);

        switch (zret) {
          case Z_NEED_DICT:
          case Z_DATA_ERROR:
          case Z_MEM_ERROR:
            inflateEnd(&zstr);
            return UnZip::WriteFailed;
          default:
            ;
        }

        szDecomp = UNZIP_READ_BUFFER - zstr.avail_out;
        if (dev->write(buffer2, szDecomp) != szDecomp)
        {
          inflateEnd(&zstr);
          return UnZip::ZlibError;
        }

        myCRC = crc32(myCRC, (const Bytef*) buffer2, szDecomp);

      } while (zstr.avail_out == 0);

    }
    while (zret != Z_STREAM_END);

    inflateEnd(&zstr);
  }

  if (myCRC != entry.crc)
    return UnZip::Corrupted;

  return UnZip::Ok;
}
```

Here is the call graph for this function:



4.45.3.8 quint64 getULLong (const unsigned char ∗ *data*, quint32 *offset*) const [inline]

Definition at line 1232 of file unzip.cpp.

```
{
  quint64 res = (quint64) data[offset];
  res |= (((quint64)data[offset+1]) << 8);
  res |= (((quint64)data[offset+2]) << 16);
  res |= (((quint64)data[offset+3]) << 24);
  res |= (((quint64)data[offset+1]) << 32);
  res |= (((quint64)data[offset+2]) << 40);
  res |= (((quint64)data[offset+3]) << 48);
  res |= (((quint64)data[offset+3]) << 56);

  return res;
}
```

4.45.3.9 quint32 getULong (const unsigned char ∗ *data,* quint32 *offset*) const [inline]

Definition at line 1219 of file unzip.cpp.

```
{
  quint32 res = (quint32) data[offset];
  res |= (((quint32)data[offset+1]) << 8);
  res |= (((quint32)data[offset+2]) << 16);
  res |= (((quint32)data[offset+3]) << 24);

  return res;
}
```

4.45.3.10 quint16 getUShort (const unsigned char ∗ *data,* quint32 *offset*) const [inline]

Definition at line 1249 of file unzip.cpp.

```
{
  return (quint16) data[offset] | (((quint16)data[offset+1]) << 8);
}
```

4.45.3.11 void initKeys (const QString & *pwd,* quint32 ∗ *keys*) const [inline]

Definition at line 1278 of file unzip.cpp.

```
{
  keys[0] = 305419896L;
  keys[1] = 591751049L;
  keys[2] = 878082192L;

    //QByteArray pwdBytes = pwd.toAscii(); // Qt4
    QByteArray pwdBytes = pwd.toLatin1();  // Qt5
  int sz = pwdBytes.size();
  const char* ascii = pwdBytes.data();

  for (int i=0; i<sz; ++i)
    updateKeys(keys, (int)ascii[i]);
}
```

4.45.3.12 UnZip::ErrorCode openArchive (QIODevice ∗ *device*)

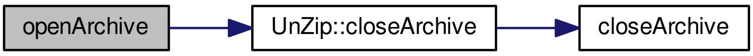
Todo Ignore CD entry count? CD may be corrupted.

Definition at line 500 of file unzip.cpp.

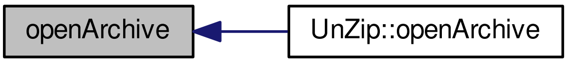
Referenced by UnZip::openArchive().

```
{
  Q_ASSERT(dev != 0);

  if (device != 0)
    closeArchive();

  device = dev;

  if (!(device->isOpen() || device->open(QIODevice::ReadOnly)))
  {
    delete device;
    device = 0;

    qDebug() << "Unable to open device for reading";
    return UnZip::OpenFailed;
  }

  UnZip::ErrorCode ec;

  ec = seekToCentralDirectory();
  if (ec != UnZip::Ok)
  {
    closeArchive();
    return ec;
  }

  //! \todo Ignore CD entry count? CD may be corrupted.
  if (cdEntryCount == 0)
  {
    return UnZip::Ok;
  }

  bool continueParsing = true;

  while (continueParsing)
  {
    if (device->read(buffer1, 4) != 4)
      UNZIP_CHECK_FOR_VALID_DATA

    if (! (buffer1[0] == 'P' && buffer1[1] == 'K' && buffer1[2] == 0x01  &&
buffer1[3] == 0x02) )
      break;

    if ( (ec = parseCentralDirectoryRecord()) !=
UnZip::Ok )
      break;
  }

  if (ec != UnZip::Ok)
    closeArchive();

  return ec;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.45.3.13 UnZip::ErrorCode parseCentralDirectoryRecord ( )

Definition at line 837 of file unzip.cpp.

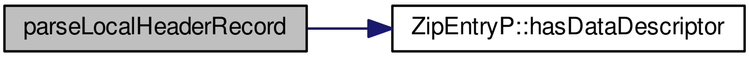
```
{
  // Read CD record
  if (device->read(buffer1, UNZIP_CD_ENTRY_SIZE_NS) != UNZIP_CD_ENTRY_SIZE_NS)
    return UnZip::ReadFailed;

  bool skipEntry = false;

  // Get compression type so we can skip non compatible algorithms
  quint16 compMethod = getUShort(uBuffer, UNZIP_CD_OFF_CMETHOD);

  // Get variable size fields length so we can skip the whole record
  // if necessary
  quint16 szName = getUShort(uBuffer, UNZIP_CD_OFF_NAMELEN);
  quint16 szExtra = getUShort(uBuffer, UNZIP_CD_OFF_XLEN);
  quint16 szComment = getUShort(uBuffer, UNZIP_CD_OFF_COMMLEN);

  quint32 skipLength = szName + szExtra + szComment;

  UnZip::ErrorCode ec = UnZip::Ok;

  if ((compMethod != 0) && (compMethod != 8))
  {
    qDebug() << "Unsupported compression method. Skipping file.";
    skipEntry = true;
  }

  // Header parsing may be a problem if version is bigger than UNZIP_VERSION
  if (!skipEntry && buffer1[UNZIP_CD_OFF_VERSION] > UNZIP_VERSION)
  {
    qDebug() << "Unsupported PKZip version. Skipping file.";
    skipEntry = true;
  }

  if (!skipEntry && szName == 0)
  {
    qDebug() << "Skipping file with no name.";
    skipEntry = true;
  }

  if (!skipEntry && device->read(buffer2, szName) != szName)
  {
    ec = UnZip::ReadFailed;
    skipEntry = true;
  }

  if (skipEntry)
  {
    if (ec == UnZip::Ok)
    {
      if (!device->seek( device->pos() + skipLength ))
        ec = UnZip::SeekFailed;

      unsupportedEntryCount++;
    }

    return ec;
  }

    //QString filename = QString::fromAscii(buffer2, szName); // Qt4
    QString filename = QString::fromLatin1(buffer2, szName);  // Qt5

  ZipEntryP* h = new ZipEntryP;
  h->compMethod = compMethod;

  h->gpFlag[0] = buffer1[UNZIP_CD_OFF_GPFLAG];
  h->gpFlag[1] = buffer1[UNZIP_CD_OFF_GPFLAG + 1];

  h->modTime[0] = buffer1[UNZIP_CD_OFF_MODT];
  h->modTime[1] = buffer1[UNZIP_CD_OFF_MODT + 1];

  h->modDate[0] = buffer1[UNZIP_CD_OFF_MODD];
  h->modDate[1] = buffer1[UNZIP_CD_OFF_MODD + 1];

  h->crc = getULong(uBuffer, UNZIP_CD_OFF_CRC32);
  h->szComp = getULong(uBuffer, UNZIP_CD_OFF_CSIZE);
  h->szUncomp = getULong(uBuffer, UNZIP_CD_OFF_USIZE);

  // Skip extra field (if any)
  if (szExtra != 0)
  {
    if (!device->seek( device->pos() + szExtra ))
    {
      delete h;
      return UnZip::SeekFailed;
    }
  }

  // Read comment field (if any)
  if (szComment != 0)
  {
    if (device->read(buffer2, szComment) != szComment)
    {
      delete h;
      return UnZip::ReadFailed;
    }

        //h->comment = QString::fromAscii(buffer2, szComment); // Qt4
        h->comment = QString::fromLatin1(buffer2, szComment);  // Qt5
  }

  h->lhOffset = getULong(uBuffer, UNZIP_CD_OFF_LHOFFSET);

  if (headers == 0)
    headers = new QMap<QString, ZipEntryP*>();
  headers->insert(filename, h);

  return UnZip::Ok;
}
```

4.45.3.14 UnZip::ErrorCode parseLocalHeaderRecord (const QString & *path*, ZipEntryP & *entry*)

Definition at line 574 of file unzip.cpp.

```
{
  if (!device->seek(entry.lhOffset))
    return UnZip::SeekFailed;

  // Test signature
  if (device->read(buffer1, 4) != 4)
    return UnZip::ReadFailed;

  if ((buffer1[0] != 'P') || (buffer1[1] != 'K') || (buffer1[2] != 0x03) || (buffer1[3] != 0x04))
    return UnZip::InvalidArchive;

  if (device->read(buffer1, UNZIP_LOCAL_HEADER_SIZE) != UNZIP_LOCAL_HEADER_SIZE)
    return UnZip::ReadFailed;

  /*
    Check 3rd general purpose bit flag.

    "bit 3: If this bit is set, the fields crc-32, compressed size
    and uncompressed size are set to zero in the local
    header.  The correct values are put in the data descriptor
    immediately following the compressed data."
  */
  bool hasDataDescriptor = entry.hasDataDescriptor();

  bool checkFailed = false;

  if (!checkFailed)
    checkFailed = entry.compMethod != getUShort(uBuffer, UNZIP_LH_OFF_CMETHOD);
  if (!checkFailed)
    checkFailed = entry.gpFlag[0] != uBuffer[UNZIP_LH_OFF_GPFLAG];
  if (!checkFailed)
    checkFailed = entry.gpFlag[1] != uBuffer[UNZIP_LH_OFF_GPFLAG + 1];
  if (!checkFailed)
    checkFailed = entry.modTime[0] != uBuffer[UNZIP_LH_OFF_MODT];
  if (!checkFailed)
    checkFailed = entry.modTime[1] != uBuffer[UNZIP_LH_OFF_MODT + 1];
  if (!checkFailed)
    checkFailed = entry.modDate[0] != uBuffer[UNZIP_LH_OFF_MODD];
  if (!checkFailed)
    checkFailed = entry.modDate[1] != uBuffer[UNZIP_LH_OFF_MODD + 1];
  if (!hasDataDescriptor)
  {
    if (!checkFailed)
      checkFailed = entry.crc != getULong(uBuffer, UNZIP_LH_OFF_CRC32);
    if (!checkFailed)
      checkFailed = entry.szComp != getULong(uBuffer, UNZIP_LH_OFF_CSIZE);
    if (!checkFailed)
      checkFailed = entry.szUncomp != getULong(uBuffer,
UNZIP_LH_OFF_USIZE);
  }

  if (checkFailed)
    return UnZip::HeaderConsistencyError;

  // Check filename
  quint16 szName = getUShort(uBuffer, UNZIP_LH_OFF_NAMELEN);
  if (szName == 0)
    return UnZip::HeaderConsistencyError;

  if (device->read(buffer2, szName) != szName)
    return UnZip::ReadFailed;

    //QString filename = QString::fromAscii(buffer2, szName); // Qt4
    QString filename = QString::fromLatin1(buffer2, szName);  // Qt5
  if (filename != path)
  {
    qDebug() << "Filename in local header mismatches.";
    return UnZip::HeaderConsistencyError;
  }

  // Skip extra field
  quint16 szExtra = getUShort(uBuffer, UNZIP_LH_OFF_XLEN);
  if (szExtra != 0)
  {
    if (!device->seek(device->pos() + szExtra))
      return UnZip::SeekFailed;
  }

  entry.dataOffset = device->pos();

  if (hasDataDescriptor)
  {
    /*
      The data descriptor has this OPTIONAL signature: PK\7\8
      We try to skip the compressed data relying on the size set in the
      Central Directory record.
    */
    if (!device->seek(device->pos() + entry.szComp))
      return UnZip::SeekFailed;

    // Read 4 bytes and check if there is a data descriptor signature
    if (device->read(buffer2, 4) != 4)
      return UnZip::ReadFailed;

    bool hasSignature = buffer2[0] == 'P' && buffer2[1] == 'K' &&
buffer2[2] == 0x07 && buffer2[3] == 0x08;
    if (hasSignature)
    {
      if (device->read(buffer2, UNZIP_DD_SIZE) !=
UNZIP_DD_SIZE)
        return UnZip::ReadFailed;
    }
    else
    {
      if (device->read(buffer2 + 4, UNZIP_DD_SIZE - 4) !=
UNZIP_DD_SIZE - 4)
        return UnZip::ReadFailed;
    }

    // DD: crc, compressed size, uncompressed size
    if (
      entry.crc != getULong((unsigned char*)buffer2,
UNZIP_DD_OFF_CRC32) ||
      entry.szComp != getULong((unsigned char*)buffer2,
UNZIP_DD_OFF_CSIZE) ||
      entry.szUncomp != getULong((unsigned char*)buffer2,
UNZIP_DD_OFF_USIZE)
      )
      return UnZip::HeaderConsistencyError;
  }

  return UnZip::Ok;
}
```

Here is the call graph for this function:



4.45.3.15 UnZip::ErrorCode seekToCentralDirectory ( )

Definition at line 713 of file unzip.cpp.

```
{
  qint64 length = device->size();
  qint64 offset = length - UNZIP_EOCD_SIZE;

  if (length < UNZIP_EOCD_SIZE)
    return UnZip::InvalidArchive;

  if (!device->seek( offset ))
    return UnZip::SeekFailed;

  if (device->read(buffer1, UNZIP_EOCD_SIZE) != UNZIP_EOCD_SIZE)
    return UnZip::ReadFailed;

  bool eocdFound = (buffer1[0] == 'P' && buffer1[1] == 'K' &&
buffer1[2] == 0x05 && buffer1[3] == 0x06);

  if (eocdFound)
  {
    // Zip file has no comment (the only variable length field in the EOCD record)
    eocdOffset = offset;
  }
  else
  {
    qint64 read;
    char* p = 0;

    offset -= UNZIP_EOCD_SIZE;

    if (offset <= 0)
      return UnZip::InvalidArchive;

    if (!device->seek( offset ))
      return UnZip::SeekFailed;

    while ((read = device->read(buffer1, UNZIP_EOCD_SIZE)) >= 0)
    {
      if ( (p = strstr(buffer1, "PK\5\6")) != 0)
      {
        // Seek to the start of the EOCD record so we can read it fully
        // Yes... we could simply read the missing bytes and append them to the buffer
        // but this is far easier so heck it!
        device->seek( offset + (p - buffer1) );
        eocdFound = true;
        eocdOffset = offset + (p - buffer1);

        // Read EOCD record
        if (device->read(buffer1, UNZIP_EOCD_SIZE) !=
UNZIP_EOCD_SIZE)
          return UnZip::ReadFailed;

        break;
      }

      offset -= UNZIP_EOCD_SIZE;
      if (offset <= 0)
        return UnZip::InvalidArchive;

      if (!device->seek( offset ))
        return UnZip::SeekFailed;
    }
  }

  if (!eocdFound)
    return UnZip::InvalidArchive;

  // Parse EOCD to locate CD offset
  offset = getULong((const unsigned char*)buffer1, UNZIP_EOCD_OFF_CDOFF + 4);

  cdOffset = offset;

  cdEntryCount = getUShort((const unsigned char*)buffer1, UNZIP_EOCD_OFF_ENTRIES + 4);

  quint16 commentLength = getUShort((const unsigned char*)buffer1, UNZIP_EOCD_OFF_COMMLEN + 4);
  if (commentLength != 0)
  {
    QByteArray c = device->read(commentLength);
    if (c.count() != commentLength)
      return UnZip::ReadFailed;

    comment = c;
  }

  // Seek to the start of the CD record
  if (!device->seek( cdOffset ))
    return UnZip::SeekFailed;

  return UnZip::Ok;
}
```

4.45.3.16 bool testKeys (const ZipEntryP & *header,* quint32 ∗ *keys*)

Definition at line 1317 of file unzip.cpp.

```
{
  char lastByte;

  // decrypt encryption header
  for (int i=0; i<11; ++i)
    updateKeys(keys, lastByte = buffer1[i] ^ decryptByte(keys[2]));
  updateKeys(keys, lastByte = buffer1[11] ^ decryptByte(keys[2]));

  // if there is an extended header (bit in the gp flag) buffer[11] is a byte from the file time
  // with no extended header we have to check the crc high-order byte
  char c = ((header.gpFlag[0] & 0x08) == 8) ? header.modTime[1] : header.crc >> 24;

  return (lastByte == c);
}
```

4.45.3.17 UnZip::ErrorCode testPassword (quint32 ∗ *keys,* const QString & *file,* const ZipEntryP & *header*)

Definition at line 1298 of file unzip.cpp.

```
{
  Q_UNUSED(file);

  // read encryption keys
  if (device->read(buffer1, 12) != 12)
    return UnZip::Corrupted;

  // Replace this code if you want to i.e. call some dialog and ask the user for a password
  initKeys(password, keys);
  if (testKeys(header, keys))
    return UnZip::Ok;

  return UnZip::Skip;
}
```

4.45.3.18 void updateKeys (quint32 ∗ *keys,* int *c*) const `[inline]`

Definition at line 1266 of file unzip.cpp.

```
{
  keys[0] = CRC32(keys[0], c);
  keys[1] += keys[0] & 0xff;
  keys[1] = keys[1] * 134775813L + 1;
  keys[2] = CRC32(keys[2], ((int)keys[1]) >> 24);
}
```

4.45.4 Member Data Documentation

4.45.4.1 char buffer1[UNZIP_READ_BUFFER]

Definition at line 65 of file unzip_p.h.

4.45.4.2 char buffer2[UNZIP_READ_BUFFER]

Definition at line 66 of file unzip_p.h.

4.45.4.3 quint16 cdEntryCount

Definition at line 77 of file unzip_p.h.

4.45.4.4 quint32 cdOffset

Definition at line 72 of file unzip_p.h.

4.45.4.5 QString comment

Definition at line 82 of file unzip_p.h.

Referenced by UnZip::archiveComment().

4.45.4.6 const quint32∗ crcTable

Definition at line 69 of file unzip_p.h.

4.45.4.7 QIODevice∗ device

Definition at line 63 of file unzip_p.h.

Referenced by UnZip::archiveComment(), UnZip::extractAll(), and UnZip::isOpen().

4.45.4.8 quint32 eocdOffset

Definition at line 74 of file unzip_p.h.

4.45.4.9 QMap<QString,ZipEntryP∗>∗ headers

Definition at line 61 of file unzip_p.h.

Referenced by UnZip::contains(), UnZip::entryList(), UnZip::extractAll(), UnZip::extractFile(), and UnZip::fileList().

4.45.4.10 QString password

Definition at line 57 of file unzip_p.h.

Referenced by UnZip::setPassword().

4.45.4.11 bool skipAllEncrypted

Definition at line 59 of file unzip_p.h.

Referenced by UnZip::extractAll().

4.45.4.12 unsigned char∗ uBuffer

Definition at line 68 of file unzip_p.h.

4.45.4.13 quint16 unsupportedEntryCount

Definition at line 80 of file unzip_p.h.

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

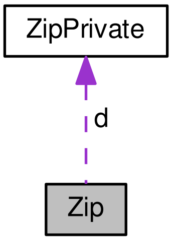
- /home/lobianco/git/ffsm_pp/src/unzip_p.h
- /home/lobianco/git/ffsm_pp/src/unzip.cpp

4.46 Zip Class Reference

Zip file compression.

`#include <zip.h>`

Collaboration diagram for Zip:



Public Types

- enum ErrorCode {
  Ok, ZlibInit, ZlibError, FileExists,
  OpenFailed, NoOpenArchive, FileNotFound, ReadFailed,
  WriteFailed, SeekFailed }
- enum CompressionLevel {
  Store, Deflate1 = 1, Deflate2, Deflate3,
  Deflate4, Deflate5, Deflate6, Deflate7,
  Deflate8, Deflate9, AutoCPU, AutoMIME,
  AutoFull }
- enum CompressionOption { RelativePaths = 0x0001, AbsolutePaths = 0x0002, IgnorePaths = 0x0004 }

Public Member Functions

- Zip ()
- virtual ∼Zip ()
- bool isOpen () const
- void setPassword (const QString &pwd)
- void clearPassword ()

  Convenience method, clears the current password.
- QString password () const

  Returns the currently used password.
- ErrorCode createArchive (const QString &file, bool overwrite=true)
- ErrorCode createArchive (QIODevice ∗device)
- QString archiveComment () const
- void setArchiveComment (const QString &comment)
- ErrorCode addDirectoryContents (const QString &path, CompressionLevel level=AutoFull)
- ErrorCode addDirectoryContents (const QString &path, const QString &root, CompressionLevel level=AutoFull)
- ErrorCode addDirectory (const QString &path, CompressionOptions options=RelativePaths, CompressionLevel level=AutoFull)
- ErrorCode addDirectory (const QString &path, const QString &root, CompressionLevel level=AutoFull)
- ErrorCode addDirectory (const QString &path, const QString &root, CompressionOptions options=RelativePaths, CompressionLevel level=AutoFull)
- ErrorCode closeArchive ()
- QString formatError (ErrorCode c) const

Private Attributes

- ZipPrivate ∗ d

4.46.1 Detailed Description

Zip file compression.

Some quick usage examples.

```
Suppose you have this directory structure:

/root/dir1/
/root/dir1/file1.1
/root/dir1/file1.2
/root/dir1/dir1.1/
/root/dir1/dir1.2/file1.2.1

EXAMPLE 1:
myZipInstance.addDirectory("/root/dir1");

RESULT:
Beheaves like any common zip software and creates a zip file with this structure:

dir1/
dir1/file1.1
dir1/file1.2
dir1/dir1.1/
dir1/dir1.2/file1.2.1

EXAMPLE 2:
myZipInstance.addDirectory("/root/dir1", "myRoot/myFolder");

RESULT:
Adds a custom root to the paths and creates a zip file with this structure:

myRoot/myFolder/dir1/
myRoot/myFolder/dir1/file1.1
myRoot/myFolder/dir1/file1.2
myRoot/myFolder/dir1/dir1.1/
myRoot/myFolder/dir1/dir1.2/file1.2.1

EXAMPLE 3:
myZipInstance.addDirectory("/root/dir1", Zip::AbsolutePaths);

NOTE:
Same as calling addDirectory(SOME_PATH, PARENT_PATH_of_SOME_PATH).

RESULT:
Preserves absolute paths and creates a zip file with this structure:

/root/dir1/
/root/dir1/file1.1
/root/dir1/file1.2
/root/dir1/dir1.1/
/root/dir1/dir1.2/file1.2.1

EXAMPLE 4:
myZipInstance.setPassword("hellopass");
myZipInstance.addDirectory("/root/dir1", "/");

RESULT:
Adds and encrypts the files in /root/dir1, creating the following zip structure:

/dir1/
/dir1/file1.1
/dir1/file1.2
/dir1/dir1.1/
/dir1/dir1.2/file1.2.1
```

Definition at line 45 of file zip.h.

4.46.2 Member Enumeration Documentation

4.46.2.1 enum CompressionLevel

Returns the result of a decompression operation. Zip::Store No compression. Zip::Deflate1 Deflate compression level 1(lowest compression). Zip::Deflate1 Deflate compression level 2. Zip::Deflate1 Deflate compression level

3. Zip::Deflate1 Deflate compression level 4. Zip::Deflate1 Deflate compression level 5. Zip::Deflate1 Deflate compression level 6. Zip::Deflate1 Deflate compression level 7. Zip::Deflate1 Deflate compression level 8. Zip::Deflate1 Deflate compression level 9 (maximum compression). Zip::AutoCPU Adapt compression level to CPU speed (faster CPU => better compression). Zip::AutoMIME Adapt compression level to MIME type of the file being compressed. Zip::AutoFull Use both CPU and MIME type detection.

Enumerator

Store

Deflate1

Deflate2

Deflate3

Deflate4

Deflate5

Deflate6

Deflate7

Deflate8

Deflate9

AutoCPU

AutoMIME

AutoFull

Definition at line 62 of file zip.h.

```
  {
    Store,
    Deflate1 = 1, Deflate2, Deflate3, Deflate4,
    Deflate5, Deflate6, Deflate7, Deflate8,
Deflate9,
    AutoCPU, AutoMIME, AutoFull
  };
```

4.46.2.2 enum CompressionOption

Enumerator

RelativePaths Does not preserve absolute paths in the zip file when adding a file/directory (default)

AbsolutePaths Preserve absolute paths.

IgnorePaths Do not store paths. All the files are put in the (evtl. user defined) root of the zip file.

Definition at line 70 of file zip.h.

```
  {
    //! Does not preserve absolute paths in the zip file when adding a file/directory (default)
    RelativePaths = 0x0001,
    //! Preserve absolute paths
    AbsolutePaths = 0x0002,
    //! Do not store paths. All the files are put in the (evtl. user defined) root of the zip file
    IgnorePaths = 0x0004
  };
```

4.46.2.3 enum ErrorCode

The result of a compression operation. Zip::Ok No error occurred. Zip::ZlibInit Failed to init or load the zlib library. Zip::ZlibError The zlib library returned some error. Zip::FileExists The file already exists and will not be overwritten. Zip::OpenFailed Unable to create or open a device. Zip::NoOpenArchive CreateArchive() has not been called yet. Zip::FileNotFound File or directory does not exist. Zip::ReadFailed Reading of a file failed. Zip::WriteFailed Writing of a file failed. Zip::SeekFailed Seek failed.

Enumerator

- ***Ok***
- ***ZlibInit***
- ***ZlibError***
- ***FileExists***
- ***OpenFailed***
- ***NoOpenArchive***
- ***FileNotFound***
- ***ReadFailed***
- ***WriteFailed***
- ***SeekFailed***

Definition at line 48 of file zip.h.

```
00049   {
00050     Ok,
00051     ZlibInit,
00052     ZlibError,
00053     FileExists,
00054     OpenFailed,
00055     NoOpenArchive,
00056     FileNotFound,
00057     ReadFailed,
00058     WriteFailed,
00059     SeekFailed
00060   };
```

4.46.3 Constructor & Destructor Documentation

4.46.3.1 Zip ( )

Creates a new Zip file compressor.

Definition at line 218 of file zip.cpp.

```
00219 {
00220   d = new ZipPrivate;
00221 }
```

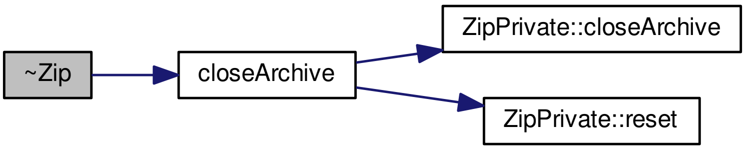
4.46.3.2 ∼Zip ( ) `[virtual]`

Closes any open archive and releases used resources.

Definition at line 226 of file zip.cpp.

```
00227 {
00228   closeArchive();
00229   delete d;
00230 }
```

Here is the call graph for this function:



4.46.4 Member Function Documentation

4.46.4.1 Zip::ErrorCode addDirectory (const QString & *path*, CompressionOptions *options* = RelativePaths, CompressionLevel *level* = AutoFull)

Convenience method, same as calling Zip::addDirectory(const QString&,const QString&,CompressionLevel) with an empty `root` parameter (or with the parent directory of `path` if the AbsolutePaths options is set).

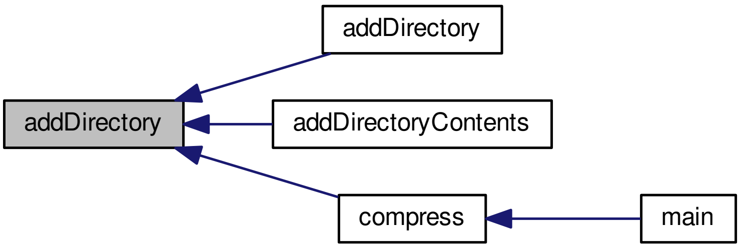
The ExtractionOptions are checked in the order they are defined in the zip.h heaser file. This means that the last one overwrites the previous one (if some conflict occurs), i.e. Zip::IgnorePaths | Zip::AbsolutePaths would be interpreted as Zip::IgnorePaths.

Definition at line 333 of file zip.cpp.

Referenced by addDirectory(), addDirectoryContents(), and compress().

```
00334 {
00335   return addDirectory(path, QString(), options, level);
00336 }
```

Here is the caller graph for this function:



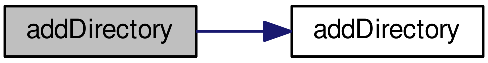
4.46.4.2 Zip::ErrorCode addDirectory (const QString & *path*, const QString & *root*, CompressionLevel *level* = AutoFull)

Convenience method, same as calling Zip::addDirectory(const QString&,const QString&,CompressionOptions,CompressionLevel) with the Zip::RelativePaths flag as compression option.

Definition at line 342 of file zip.cpp.

```
{
  return addDirectory(path, root, Zip::RelativePaths, level);
}
```

Here is the call graph for this function:



4.46.4.3 Zip::ErrorCode addDirectory (const QString & *path*, const QString & *root*, CompressionOptions *options* = RelativePaths, CompressionLevel *level* = AutoFull)

Recursively adds files contained in `dir` to the archive, using `root` as name for the root folder. Stops adding files if some error occurs.

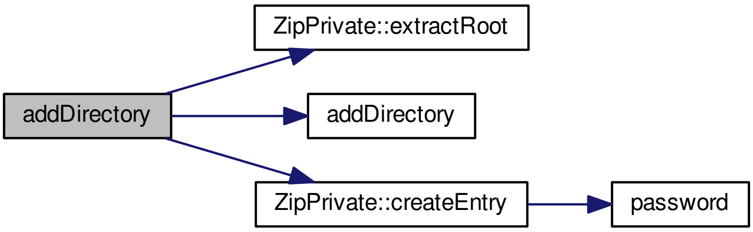
The ExtractionOptions are checked in the order they are defined in the zip.h heaser file. This means that the last one overwrites the previous one (if some conflict occurs), i.e. Zip::IgnorePaths | Zip::AbsolutePaths would be interpreted as Zip::IgnorePaths.

The `root` parameter is ignored with the Zip::IgnorePaths parameter and used as path prefix (a trailing / is always added as directory separator!) otherwise (even with Zip::AbsolutePaths set!).

Definition at line 376 of file zip.cpp.

```
{
  // qDebug() << QString("addDir(path=%1, root=%2)").arg(path, root);

  // Bad boy didn’t call createArchive() yet :)
  if (d->device == 0)
    return Zip::NoOpenArchive;

  QDir dir(path);
  if (!dir.exists())
    return Zip::FileNotFound;

  // Remove any trailing separator
  QString actualRoot = root.trimmed();

  // Preserve Unix root
  if (actualRoot != "/")
  {
    while (actualRoot.endsWith("/") || actualRoot.endsWith("\\"))
      actualRoot.truncate(actualRoot.length() - 1);
  }

  // QDir::cleanPath() fixes some issues with QDir::dirName()
  QFileInfo current(QDir::cleanPath(path));

  if (!actualRoot.isEmpty() && actualRoot != "/")
    actualRoot.append("/");

  /* This part is quite confusing and needs some test or check */
  /* An attempt to compress the / root directory evtl. using a root prefix should be a good test */
  if (options.testFlag(AbsolutePaths) && !options.testFlag(
IgnorePaths))
  {
    QString absolutePath = d->extractRoot(path);
    if (!absolutePath.isEmpty() && absolutePath != "/")
      absolutePath.append("/");
    actualRoot.append(absolutePath);
  }

  if (!options.testFlag(IgnorePaths))
  {
    actualRoot = actualRoot.append(QDir(current.absoluteFilePath()).dirName());
    actualRoot.append("/");
  }

  // actualRoot now contains the path of the file relative to the zip archive
  // with a trailing /

  QFileInfoList list = dir.entryInfoList(
    QDir::Files |
    QDir::Dirs |
    QDir::NoDotAndDotDot |
    QDir::NoSymLinks);

  ErrorCode ec = Zip::Ok;
  bool filesAdded = false;

  CompressionOptions recursionOptions;
  if (options.testFlag(IgnorePaths))
    recursionOptions |= IgnorePaths;
  else recursionOptions |= RelativePaths;

  for (int i = 0; i < list.size() && ec == Zip::Ok; ++i)
  {
    QFileInfo info = list.at(i);

    if (info.isDir())
    {
      // Recursion :)
      ec = addDirectory(info.absoluteFilePath(), actualRoot, recursionOptions, level);
    }
    else
    {
      ec = d->createEntry(info, actualRoot, level);
      filesAdded = true;
    }
  }


  // We need an explicit record for this dir
  // Non-empty directories don't need it because they have a path component in the filename
  if (!filesAdded && !options.testFlag(IgnorePaths))
    ec = d->createEntry(current, actualRoot, level);

  return ec;
}
```

Here is the call graph for this function:



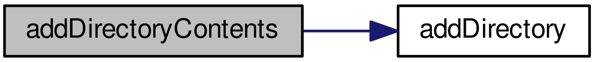
4.46.4.4 Zip::ErrorCode addDirectoryContents (const QString & *path*, CompressionLevel *level* = AutoFull)

Convenience method, same as calling Zip::addDirectory(const QString&,const QString&,CompressionOptions,CompressionLevel) with the Zip::IgnorePaths flag as compression option and an empty `root` parameter.

Definition at line 351 of file zip.cpp.

```
{
  return addDirectory(path, QString(), IgnorePaths, level);
}
```

Here is the call graph for this function:



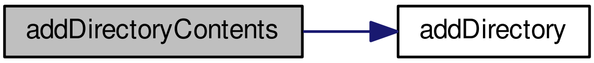
4.46.4.5 Zip::ErrorCode addDirectoryContents (const QString & *path*, const QString & *root*, CompressionLevel *level* = AutoFull)

Convenience method, same as calling Zip::addDirectory(const QString&,const QString&,CompressionOptions,CompressionLevel) with the Zip::IgnorePaths flag as compression option.

Definition at line 360 of file zip.cpp.

```
{
  return addDirectory(path, root, IgnorePaths, level);
}
```

Here is the call graph for this function:



4.46.4.6 QString archiveComment ( ) const

Returns the current archive comment.

Definition at line 308 of file zip.cpp.

```
{
  return d->comment;
}
```

4.46.4.7 void clearPassword ( )

Convenience method, clears the current password.

Definition at line 252 of file zip.cpp.

```
00253 {
00254   d->password.clear();
00255 }
```

4.46.4.8 Zip::ErrorCode closeArchive ( )

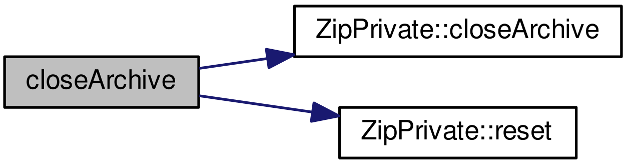
Closes the archive and writes any pending data.

Definition at line 465 of file zip.cpp.

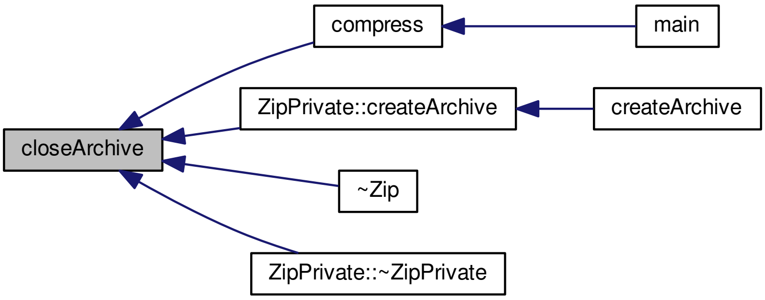
Referenced by compress(), ZipPrivate::createArchive(), ∼Zip(), and ZipPrivate::∼ZipPrivate().

```
00466 {
00467   Zip::ErrorCode ec = d->closeArchive();
00468   d->reset();
00469   return ec;
00470 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.46.4.9 Zip::ErrorCode createArchive (const QString & *filename*, bool *overwrite* = `true`)

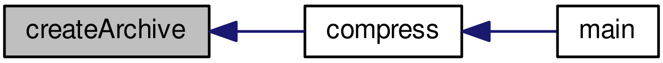
Attempts to create a new Zip archive. If `overwrite` is true and the file already exist it will be overwritten. Any open archive will be closed.

Definition at line 268 of file zip.cpp.

Referenced by compress().

```
{
  QFile* file = new QFile(filename);

  if (file->exists() && !overwrite) {
    delete file;
    return Zip::FileExists;
  }

  if (!file->open(QIODevice::WriteOnly)) {
    delete file;
    return Zip::OpenFailed;
  }

  Zip::ErrorCode ec = createArchive(file);
  if (ec != Zip::Ok) {
    file->remove();
  }

  return ec;
}
```

Here is the caller graph for this function:



4.46.4.10 Zip::ErrorCode createArchive (QIODevice ∗ *device*)

Attempts to create a new Zip archive. If there is another open archive this will be closed.

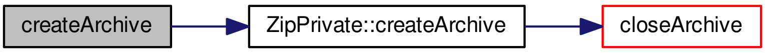
Warning

> The class takes ownership of the device!

Definition at line 294 of file zip.cpp.

```
{
  if (device == 0)
  {
    qDebug() << "Invalid device.";
    return Zip::OpenFailed;
  }

  return d->createArchive(device);
}
```

Here is the call graph for this function:



4.46.4.11 QString formatError (Zip::ErrorCode *c*) const

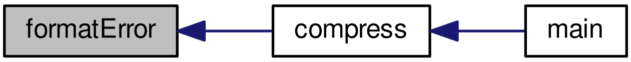
Returns a locale translated error string for a given error code.

Definition at line 475 of file zip.cpp.

Referenced by compress().

```
{
  switch (c)
  {
  case Ok: return QCoreApplication::translate("Zip", "ZIP operation completed successfully."); break;
  case ZlibInit: return QCoreApplication::translate("Zip", "Failed to initialize or load zlib library."); break;
  case ZlibError: return QCoreApplication::translate("Zip", "zlib library error."); break;
  case OpenFailed: return QCoreApplication::translate("Zip", "Unable to create or open file."); break;
  case NoOpenArchive: return QCoreApplication::translate("Zip", "No archive has been created yet."); break;
  case FileNotFound: return QCoreApplication::translate("Zip", "File or directory does not exist."); break;
  case ReadFailed: return QCoreApplication::translate("Zip", "File read error."); break;
  case WriteFailed: return QCoreApplication::translate("Zip", "File write error."); break;
  case SeekFailed: return QCoreApplication::translate("Zip", "File seek error."); break;
  default: ;
  }

  return QCoreApplication::translate("Zip", "Unknown error.");
}
```

Here is the caller graph for this function:



4.46.4.12 bool isOpen ( ) const

Returns true if there is an open archive.

Definition at line 235 of file zip.cpp.

```
{
  return d->device != 0;
}
```

4.46.4.13 QString password ( ) const

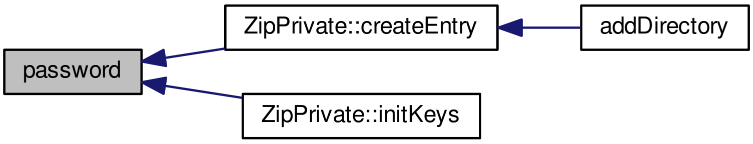
Returns the currently used password.

Definition at line 258 of file zip.cpp.

Referenced by ZipPrivate::createEntry(), and ZipPrivate::initKeys().

```
00259 {
00260   return d->password;
00261 }
```

Here is the caller graph for this function:



4.46.4.14 void setArchiveComment (const QString & *comment*)

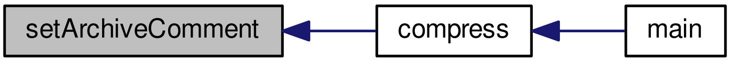
Sets the comment for this archive. Note: createArchive() should have been called before.

Definition at line 317 of file zip.cpp.

Referenced by compress().

```
00318 {
00319   if (d->device != 0)
00320     d->comment = comment;
00321 }
```

Here is the caller graph for this function:



4.46.4.15 void setPassword (const QString & *pwd*)

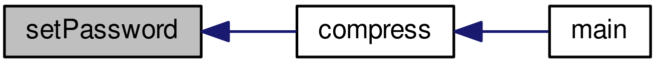
Sets the password to be used for the next files being added! Files added before calling this method will use the previously set password (if any). Closing the archive won't clear the password!

Definition at line 246 of file zip.cpp.

Referenced by compress().

```
00247 {
00248   d->password = pwd;
00249 }
```

Here is the caller graph for this function:



4.46.5 Member Data Documentation

4.46.5.1 ZipPrivate∗ d [private]

Definition at line 108 of file zip.h.

Referenced by addDirectory(), archiveComment(), clearPassword(), closeArchive(), createArchive(), ZipPrivate::createEntry(), ZipPrivate::extractRoot(), isOpen(), password(), setArchiveComment(), setPassword(), Zip(), and ∼Zip().

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

- /home/lobianco/git/ffsm_pp/src/zip.h
- /home/lobianco/git/ffsm_pp/src/zip.cpp

4.47 UnZip::ZipEntry Struct Reference

`#include <unzip.h>`

Public Member Functions

- ZipEntry ()

Public Attributes

- QString filename
- QString comment
- quint32 compressedSize
- quint32 uncompressedSize
- quint32 crc32
- QDateTime lastModified
- CompressionMethod compression
- FileType type
- bool encrypted

4.47.1 Detailed Description

Definition at line 89 of file unzip.h.

4.47.2 Constructor & Destructor Documentation

4.47.2.1 ZipEntry ( )

ZipEntry constructor - initialize data. Type is set to File.

Definition at line 471 of file unzip.cpp.

```
00472 {
00473   compressedSize = uncompressedSize = crc32 = 0;
00474   compression = NoCompression;
00475   type = File;
00476   encrypted = false;
00477 }
```

4.47.3 Member Data Documentation

4.47.3.1 QString comment

Definition at line 94 of file unzip.h.

Referenced by UnZip::entryList().

4.47.3.2 quint32 compressedSize

Definition at line 96 of file unzip.h.

Referenced by UnZip::entryList(), and listFiles().

4.47.3.3 CompressionMethod compression

Definition at line 102 of file unzip.h.

Referenced by UnZip::entryList().

4.47.3.4 quint32 crc32

Definition at line 98 of file unzip.h.

Referenced by UnZip::entryList(), and listFiles().

4.47.3.5 bool encrypted

Definition at line 105 of file unzip.h.

Referenced by UnZip::entryList(), and listFiles().

4.47.3.6 QString filename

Definition at line 93 of file unzip.h.

Referenced by UnZip::entryList(), and listFiles().

4.47.3.7 QDateTime lastModified

Definition at line 100 of file unzip.h.

Referenced by UnZip::entryList().

4.47.3.8 FileType type

Definition at line 103 of file unzip.h.

Referenced by UnZip::entryList().

4.47.3.9 quint32 uncompressedSize

Definition at line 97 of file unzip.h.

Referenced by UnZip::entryList(), and listFiles().

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

- /home/lobianco/git/ffsm_pp/src/unzip.h
- /home/lobianco/git/ffsm_pp/src/unzip.cpp

4.48 ZipEntryP Class Reference

`#include <zipentry_p.h>`

Public Member Functions

- ZipEntryP ()
- bool isEncrypted () const
- bool hasDataDescriptor () const

Public Attributes

- quint32 lhOffset
- quint32 dataOffset
- unsigned char gpFlag [2]
- quint16 compMethod
- unsigned char modTime [2]
- unsigned char modDate [2]
- quint32 crc
- quint32 szComp
- quint32 szUncomp
- QString comment
- bool lhEntryChecked

4.48.1 Detailed Description

Definition at line 45 of file zipentry_p.h.

4.48.2 Constructor & Destructor Documentation

4.48.2.1 ZipEntryP ( ) [inline]

Definition at line 48 of file zipentry_p.h.

```
00049   {
00050     lhOffset = 0;
00051     dataOffset = 0;
00052     gpFlag[0] = gpFlag[1] = 0;
00053     compMethod = 0;
00054     modTime[0] = modTime[1] = 0;
00055     modDate[0] = modDate[1] = 0;
00056     crc = 0;
00057     szComp = szUncomp = 0;
00058     lhEntryChecked = false;
00059   }
```

4.48.3 Member Function Documentation

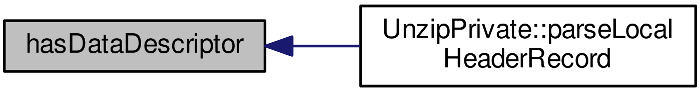
4.48.3.1 bool hasDataDescriptor ( ) const [inline]

Definition at line 75 of file zipentry_p.h.

Referenced by UnzipPrivate::parseLocalHeaderRecord().

```
00075 { return gpFlag[0] & 0x08; }
```

Here is the caller graph for this function:



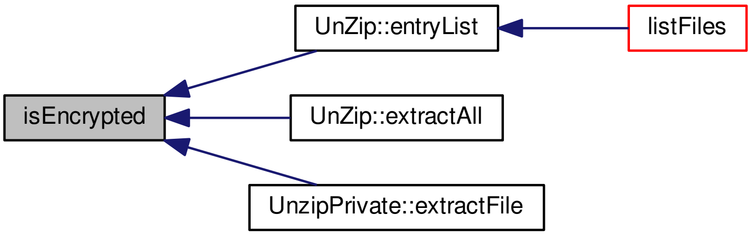
4.48.3.2 bool isEncrypted ( ) const [inline]

Definition at line 74 of file zipentry_p.h.

Referenced by UnZip::entryList(), UnZip::extractAll(), and UnzipPrivate::extractFile().

```
00074 { return gpFlag[0] & 0x01; }
```

Here is the caller graph for this function:



4.48.4 Member Data Documentation

4.48.4.1 QString comment

Definition at line 70 of file zipentry_p.h.

Referenced by UnZip::entryList(), and UnzipPrivate::parseCentralDirectoryRecord().

4.48.4.2 quint16 compMethod

Definition at line 64 of file zipentry_p.h.

Referenced by ZipPrivate::closeArchive(), ZipPrivate::createEntry(), UnZip::entryList(), UnzipPrivate::extractFile(), UnzipPrivate::parseCentralDirectoryRecord(), UnzipPrivate::parseLocalHeaderRecord(), and ZipEntryP().

4.48.4.3 quint32 crc

Definition at line 67 of file zipentry_p.h.

Referenced by ZipPrivate::closeArchive(), ZipPrivate::createEntry(), UnZip::entryList(), UnzipPrivate::extractFile(), UnzipPrivate::parseCentralDirectoryRecord(), UnzipPrivate::parseLocalHeaderRecord(), UnzipPrivate::testKeys(), and ZipEntryP().

4.48.4.4 quint32 dataOffset

Definition at line 62 of file zipentry_p.h.

Referenced by UnzipPrivate::extractFile(), UnzipPrivate::parseLocalHeaderRecord(), and ZipEntryP().

4.48.4.5 unsigned char gpFlag[2]

Definition at line 63 of file zipentry_p.h.

Referenced by ZipPrivate::closeArchive(), ZipPrivate::createEntry(), UnzipPrivate::parseCentralDirectoryRecord(), UnzipPrivate::parseLocalHeaderRecord(), UnzipPrivate::testKeys(), and ZipEntryP().

4.48.4.6 bool lhEntryChecked

Definition at line 72 of file zipentry_p.h.

Referenced by UnzipPrivate::extractFile(), and ZipEntryP().

4.48.4.7 quint32 lhOffset

Definition at line 61 of file zipentry_p.h.

Referenced by ZipPrivate::closeArchive(), ZipPrivate::createEntry(), UnzipPrivate::parseCentralDirectoryRecord(), UnzipPrivate::parseLocalHeaderRecord(), and ZipEntryP().

4.48.4.8 unsigned char modDate[2]

Definition at line 66 of file zipentry_p.h.

Referenced by ZipPrivate::closeArchive(), ZipPrivate::createEntry(), UnZip::entryList(), UnzipPrivate::parseCentralDirectoryRecord(), UnzipPrivate::parseLocalHeaderRecord(), and ZipEntryP().

4.48.4.9 unsigned char modTime[2]

Definition at line 65 of file zipentry_p.h.

Referenced by ZipPrivate::closeArchive(), ZipPrivate::createEntry(), UnZip::entryList(), UnzipPrivate::parseCentralDirectoryRecord(), UnzipPrivate::parseLocalHeaderRecord(), UnzipPrivate::testKeys(), and ZipEntryP().

4.48.4.10 quint32 szComp

Definition at line 68 of file zipentry_p.h.

Referenced by ZipPrivate::closeArchive(), ZipPrivate::createEntry(), UnZip::entryList(), UnzipPrivate::extractFile(), UnzipPrivate::parseCentralDirectoryRecord(), UnzipPrivate::parseLocalHeaderRecord(), and ZipEntryP().

4.48.4.11 quint32 szUncomp

Definition at line 69 of file zipentry_p.h.

Referenced by ZipPrivate::closeArchive(), ZipPrivate::createEntry(), UnZip::entryList(), UnzipPrivate::parseCentralDirectoryRecord(), UnzipPrivate::parseLocalHeaderRecord(), and ZipEntryP().

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

- /home/lobianco/git/ffsm_pp/src/zipentry_p.h

4.49 ZipPrivate Class Reference

`#include <zip_p.h>`

Public Member Functions

- ZipPrivate ()
- virtual ∼ZipPrivate ()
- Zip::ErrorCode createArchive (QIODevice ∗device)
- Zip::ErrorCode closeArchive ()
- void reset ()
- bool zLibInit ()
- Zip::ErrorCode createEntry (const QFileInfo &file, const QString &root, Zip::CompressionLevel level)
- Zip::CompressionLevel detectCompressionByMime (const QString &ext)
- void encryptBytes (quint32 ∗keys, char ∗buffer, qint64 read)
- void setULong (quint32 v, char ∗buffer, unsigned int offset)
- void updateKeys (quint32 ∗keys, int c) const
- void initKeys (quint32 ∗keys) const
- int decryptByte (quint32 key2) const
- QString extractRoot (const QString &p)

Public Attributes

- QMap< QString, ZipEntryP ∗ > ∗ headers
- QIODevice ∗ device
- char buffer1 [ZIP_READ_BUFFER]
- char buffer2 [ZIP_READ_BUFFER]
- unsigned char ∗ uBuffer
- const quint32 ∗ crcTable
- QString comment
- QString password

4.49.1 Detailed Description

Definition at line 54 of file zip_p.h.

4.49.2 Constructor & Destructor Documentation

4.49.2.1 ZipPrivate ( )

Definition at line 500 of file zip.cpp.

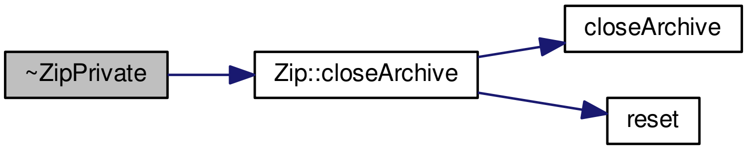
```
{
  headers = 0;
  device = 0;

  // keep an unsigned pointer so we avoid to over bloat the code with casts
  uBuffer = (unsigned char*) buffer1;
  crcTable = (quint32*) get_crc_table();
}
```

4.49.2.2 ∼ZipPrivate ( ) [virtual]

Definition at line 511 of file zip.cpp.

```
{
  closeArchive();
}
```

Here is the call graph for this function:



4.49.3 Member Function Documentation

4.49.3.1 Zip::ErrorCode closeArchive ( )

Closes the current archive and writes out pending data.

Todo See if we can detect QFile objects using the Qt Meta Object System

Todo SAME AS ABOVE: See if we can detect QFile objects using the Qt Meta Object System

Todo SAME AS ABOVE: See if we can detect QFile objects using the Qt Meta Object System

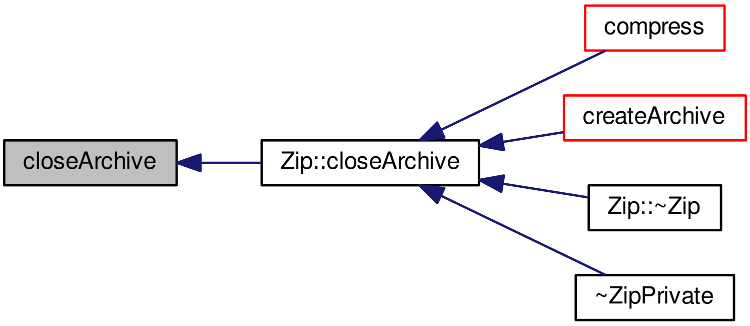
Todo SAME AS ABOVE: See if we can detect QFile objects using the Qt Meta Object System

Definition at line 1014 of file zip.cpp.

Referenced by Zip::closeArchive().

```
{
  // Close current archive by writing out central directory
  // and free up resources

  if (device == 0)
    return Zip::Ok;

  if (headers == 0)
    return Zip::Ok;

  const ZipEntryP* h;

  unsigned int sz;
  quint32 szCentralDir = 0;
  quint32 offCentralDir = device->pos();

  for (QMap<QString,ZipEntryP*>::ConstIterator itr = headers->constBegin(); itr != headers->constEnd(); ++itr)
  {
    h = itr.value();

    // signature
    buffer1[0] = 'P';
    buffer1[1] = 'K';
    buffer1[2] = 0x01;
    buffer1[3] = 0x02;

    // version made by  (currently only MS-DOS/FAT - no symlinks or other stuff supported)
    buffer1[ZIP_CD_OFF_MADEBY] = buffer1[ZIP_CD_OFF_MADEBY + 1] = 0;

    // version needed to extract
    buffer1[ZIP_CD_OFF_VERSION] = ZIP_VERSION;
    buffer1[ZIP_CD_OFF_VERSION + 1] = 0;

    // general purpose flag
    buffer1[ZIP_CD_OFF_GPFLAG] = h->gpFlag[0];
    buffer1[ZIP_CD_OFF_GPFLAG + 1] = h->gpFlag[1];

    // compression method
    buffer1[ZIP_CD_OFF_CMET] = h->compMethod & 0xFF;
    buffer1[ZIP_CD_OFF_CMET + 1] = (h->compMethod >> 8) & 0xFF;

    // last mod file time
    buffer1[ZIP_CD_OFF_MODT] = h->modTime[0];
    buffer1[ZIP_CD_OFF_MODT + 1] = h->modTime[1];

    // last mod file date
    buffer1[ZIP_CD_OFF_MODD] = h->modDate[0];
    buffer1[ZIP_CD_OFF_MODD + 1] = h->modDate[1];

    // crc (4bytes) [16,17,18,19]
    setULong(h->crc, buffer1, ZIP_CD_OFF_CRC);

    // compressed size (4bytes: [20,21,22,23])
    setULong(h->szComp, buffer1, ZIP_CD_OFF_CSIZE);

    // uncompressed size [24,25,26,27]
    setULong(h->szUncomp, buffer1, ZIP_CD_OFF_USIZE);

    // filename
        //QByteArray fileNameBytes = itr.key().toAscii();
        QByteArray fileNameBytes = itr.key().toLatin1();
    sz = fileNameBytes.size();
    buffer1[ZIP_CD_OFF_NAMELEN] = sz & 0xFF;
    buffer1[ZIP_CD_OFF_NAMELEN + 1] = (sz >> 8) & 0xFF;

    // extra field length
    buffer1[ZIP_CD_OFF_XLEN] = buffer1[ZIP_CD_OFF_XLEN + 1] = 0;

    // file comment length
    buffer1[ZIP_CD_OFF_COMMLEN] = buffer1[ZIP_CD_OFF_COMMLEN + 1] = 0;

    // disk number start
    buffer1[ZIP_CD_OFF_DISKSTART] = buffer1[ZIP_CD_OFF_DISKSTART + 1] = 0;

    // internal file attributes
    buffer1[ZIP_CD_OFF_IATTR] = buffer1[ZIP_CD_OFF_IATTR + 1] = 0;

    // external file attributes
    buffer1[ZIP_CD_OFF_EATTR] =
    buffer1[ZIP_CD_OFF_EATTR + 1] =
    buffer1[ZIP_CD_OFF_EATTR + 2] =
    buffer1[ZIP_CD_OFF_EATTR + 3] = 0;

    // relative offset of local header [42->45]
    setULong(h->lhOffset, buffer1, ZIP_CD_OFF_LHOFF);

    if (device->write(buffer1, ZIP_CD_SIZE) != ZIP_CD_SIZE)
    {
      //! \todo See if we can detect QFile objects using the Qt Meta Object System
      /*
      if (!device->remove())
        qDebug() << tr("Unable to delete corrupted archive: %1").arg(device->fileName());
      */
      return Zip::WriteFailed;
    }

  // Write out filename
  if ((unsigned int)device->write(fileNameBytes) != sz)
  {
   //! \todo SAME AS ABOVE: See if we can detect QFile objects using the Qt Meta Object System
   /*
   if (!device->remove())
    qDebug() << tr("Unable to delete corrupted archive: %1").arg(device->fileName());
    */
   return Zip::WriteFailed;
  }

  szCentralDir += (ZIP_CD_SIZE + sz);

 } // central dir headers loop


 // Write end of central directory

 // signature
 buffer1[0] = 'P';
 buffer1[1] = 'K';
 buffer1[2] = 0x05;
 buffer1[3] = 0x06;

 // number of this disk
 buffer1[ZIP_EOCD_OFF_DISKNUM] = buffer1[ZIP_EOCD_OFF_DISKNUM + 1] = 0;

 // number of disk with central directory
 buffer1[ZIP_EOCD_OFF_CDDISKNUM] = buffer1[ZIP_EOCD_OFF_CDDISKNUM + 1] = 0;

 // number of entries in this disk
 sz = headers->count();
  buffer1[ZIP_EOCD_OFF_ENTRIES] = sz & 0xFF;
 buffer1[ZIP_EOCD_OFF_ENTRIES + 1] = (sz >> 8) & 0xFF;

 // total number of entries
 buffer1[ZIP_EOCD_OFF_CDENTRIES] = buffer1[ZIP_EOCD_OFF_ENTRIES];
 buffer1[ZIP_EOCD_OFF_CDENTRIES + 1] = buffer1[ZIP_EOCD_OFF_ENTRIES + 1];

 // size of central directory [12->15]
 setULong(szCentralDir, buffer1, ZIP_EOCD_OFF_CDSIZE);

 // central dir offset [16->19]
 setULong(offCentralDir, buffer1, ZIP_EOCD_OFF_CDOFF);

 // ZIP file comment length
  //QByteArray commentBytes = comment.toAscii();
  QByteArray commentBytes = comment.toLatin1();
 quint16 commentLength = commentBytes.size();

 if (commentLength == 0)
 {
  buffer1[ZIP_EOCD_OFF_COMMLEN] = buffer1[ZIP_EOCD_OFF_COMMLEN + 1] = 0;
 }
 else
 {
  buffer1[ZIP_EOCD_OFF_COMMLEN] = commentLength & 0xFF;
  buffer1[ZIP_EOCD_OFF_COMMLEN + 1] = (commentLength >> 8) & 0xFF;
 }

 if (device->write(buffer1, ZIP_EOCD_SIZE) != ZIP_EOCD_SIZE)
 {
  //! \todo SAME AS ABOVE: See if we can detect QFile objects using the Qt Meta Object System
  /*
  if (!device->remove())
   qDebug() << tr("Unable to delete corrupted archive: %1").arg(device->fileName());
   */
  return Zip::WriteFailed;
 }

 if (commentLength != 0)
 {
  if ((unsigned int)device->write(commentBytes) != commentLength)
  {
   //! \todo SAME AS ABOVE: See if we can detect QFile objects using the Qt Meta Object System
   /*
   if (!device->remove())
    qDebug() << tr("Unable to delete corrupted archive: %1").arg(device->fileName());
    */
   return Zip::WriteFailed;
  }
  }

  return Zip::Ok;
}
```

Here is the caller graph for this function:



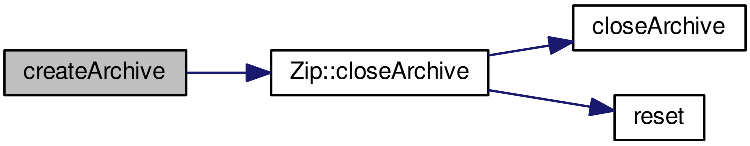
4.49.3.2 Zip::ErrorCode createArchive (QIODevice * *device*)

Definition at line 517 of file zip.cpp.

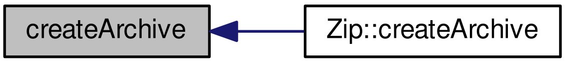
Referenced by Zip::createArchive().

```
{
  Q_ASSERT(dev != 0);

  if (device != 0)
    closeArchive();

  device = dev;

  if (!device->isOpen())
  {
    if (!device->open(QIODevice::ReadOnly)) {
      delete device;
      device = 0;
      qDebug() << "Unable to open device for writing.";
      return Zip::OpenFailed;
    }
  }

  headers = new QMap<QString,ZipEntryP*>;
  return Zip::Ok;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.49.3.3 Zip::ErrorCode createEntry (const QFileInfo & *file,* const QString & *root,* Zip::CompressionLevel *level*)

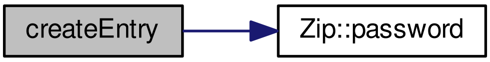
Todo Automatic level detection (cpu, extension & file size)

Definition at line 541 of file zip.cpp.

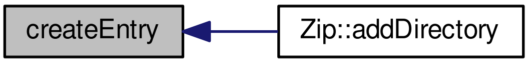
Referenced by Zip::addDirectory().

```
{
  //! \todo Automatic level detection (cpu, extension & file size)

  // Directories and very small files are always stored
  // (small files would get bigger due to the compression headers overhead)

  // Need this for zlib
  bool isPNGFile = false;
  bool dirOnly = file.isDir();

  QString entryName = root;

  // Directory entry
  if (dirOnly)
    level = Zip::Store;
  else
  {
    entryName.append(file.fileName());

    QString ext = file.completeSuffix().toLower();
    isPNGFile = ext == "png";

    if (file.size() < ZIP_COMPRESSION_THRESHOLD)
      level = Zip::Store;
    else
      switch (level)
      {
      case Zip::AutoCPU:
        level = Zip::Deflate5;
        break;
      case Zip::AutoMIME:
        level = detectCompressionByMime(ext);
        break;
      case Zip::AutoFull:
        level = detectCompressionByMime(ext);
        break;
      default:
        ;
      }
  }

  // entryName contains the path as it should be written
  // in the zip file records
  // qDebug() << QString("addDir(file=%1, root=%2, entry=%3)").arg(file.absoluteFilePath(), root, entryName);

  // create header and store it to write a central directory later
  ZipEntryP* h = new ZipEntryP;

  h->compMethod = (level == Zip::Store) ? 0 : 0x0008;

  // Set encryption bit and set the data descriptor bit
  // so we can use mod time instead of crc for password check
  bool encrypt = !dirOnly && !password.isEmpty();
  if (encrypt)
    h->gpFlag[0] |= 9;

  QDateTime dt = file.lastModified();
  QDate d = dt.date();
  h->modDate[1] = ((d.year() - 1980) << 1) & 254;
  h->modDate[1] |= ((d.month() >> 3) & 1);
  h->modDate[0] = ((d.month() & 7) << 5) & 224;
  h->modDate[0] |= d.day();

  QTime t = dt.time();
  h->modTime[1] = (t.hour() << 3) & 248;
  h->modTime[1] |= ((t.minute() >> 3) & 7);
  h->modTime[0] = ((t.minute() & 7) << 5) & 224;
  h->modTime[0] |= t.second() / 2;

  h->szUncomp = dirOnly ? 0 : file.size();

  // **** Write local file header ****

  // signature
  buffer1[0] = 'P'; buffer1[1] = 'K';
  buffer1[2] = 0x3; buffer1[3] = 0x4;

  // version needed to extract
  buffer1[ZIP_LH_OFF_VERS] = ZIP_VERSION;
  buffer1[ZIP_LH_OFF_VERS + 1] = 0;

  // general purpose flag
  buffer1[ZIP_LH_OFF_GPFLAG] = h->gpFlag[0];
  buffer1[ZIP_LH_OFF_GPFLAG + 1] = h->gpFlag[1];

  // compression method
  buffer1[ZIP_LH_OFF_CMET] = h->compMethod & 0xFF;
  buffer1[ZIP_LH_OFF_CMET + 1] = (h->compMethod>>8) & 0xFF;

  // last mod file time
  buffer1[ZIP_LH_OFF_MODT] = h->modTime[0];
  buffer1[ZIP_LH_OFF_MODT + 1] = h->modTime[1];

  // last mod file date
  buffer1[ZIP_LH_OFF_MODD] = h->modDate[0];
  buffer1[ZIP_LH_OFF_MODD + 1] = h->modDate[1];

  // skip crc (4bytes) [14,15,16,17]

  // skip compressed size but include evtl. encryption header (4bytes: [18,19,20,21])
  buffer1[ZIP_LH_OFF_CSIZE] =
  buffer1[ZIP_LH_OFF_CSIZE + 1] =
  buffer1[ZIP_LH_OFF_CSIZE + 2] =
  buffer1[ZIP_LH_OFF_CSIZE + 3] = 0;

  h->szComp = encrypt ? ZIP_LOCAL_ENC_HEADER_SIZE : 0;

  // uncompressed size [22,23,24,25]
  setULong(h->szUncomp, buffer1, ZIP_LH_OFF_USIZE);

  // filename length
    //QByteArray entryNameBytes = entryName.toAscii();
    QByteArray entryNameBytes = entryName.toLatin1(); // Qt5
  int sz = entryNameBytes.size();

  buffer1[ZIP_LH_OFF_NAMELEN] = sz & 0xFF;
  buffer1[ZIP_LH_OFF_NAMELEN + 1] = (sz >> 8) & 0xFF;

  // extra field length
  buffer1[ZIP_LH_OFF_XLEN] = buffer1[ZIP_LH_OFF_XLEN + 1] = 0;

  // Store offset to write crc and compressed size
  h->lhOffset = device->pos();
  quint32 crcOffset = h->lhOffset + ZIP_LH_OFF_CRC;

  if (device->write(buffer1, ZIP_LOCAL_HEADER_SIZE) != ZIP_LOCAL_HEADER_SIZE)
  {
    delete h;
    return Zip::WriteFailed;
  }

  // Write out filename
  if (device->write(entryNameBytes) != sz)
  {
    delete h;
    return Zip::WriteFailed;
  }

  // Encryption keys
  quint32 keys[3] = { 0, 0, 0 };

  if (encrypt)
  {
    // **** encryption header ****

    // XOR with PI to ensure better random numbers
    // with poorly implemented rand() as suggested by Info-Zip
    srand(time(NULL) ^ 3141592654UL);
    int randByte;

    initKeys(keys);
    for (int i=0; i<10; ++i)
    {
      randByte = (rand() >> 7) & 0xff;
      buffer1[i] = decryptByte(keys[2]) ^ randByte;
      updateKeys(keys, randByte);
    }

    // Encrypt encryption header
    initKeys(keys);
    for (int i=0; i<10; ++i)
    {
      randByte = decryptByte(keys[2]);
      updateKeys(keys, buffer1[i]);
      buffer1[i] ^= randByte;
    }

    // We don't know the CRC at this time, so we use the modification time
    // as the last two bytes
    randByte = decryptByte(keys[2]);
    updateKeys(keys, h->modTime[0]);
    buffer1[10] ^= randByte;

    randByte = decryptByte(keys[2]);
    updateKeys(keys, h->modTime[1]);
    buffer1[11] ^= randByte;

    // Write out encryption header
    if (device->write(buffer1, ZIP_LOCAL_ENC_HEADER_SIZE) !=
ZIP_LOCAL_ENC_HEADER_SIZE)
    {
      delete h;
      return Zip::WriteFailed;
    }
  }

  qint64 written = 0;
  quint32 crc = crc32(0L, Z_NULL, 0);

  if (!dirOnly)
  {
    QFile actualFile(file.absoluteFilePath());
    if (!actualFile.open(QIODevice::ReadOnly))
    {
      qDebug() << QString("An error occurred while opening %1").arg(file.absoluteFilePath());
      return Zip::OpenFailed;
    }

    // Write file data
    qint64 read = 0;
    qint64 totRead = 0;
    qint64 toRead = actualFile.size();

    if (level == Zip::Store)
    {
      while ( (read = actualFile.read(buffer1, ZIP_READ_BUFFER)) > 0 )
      {
        crc = crc32(crc, uBuffer, read);

        if (password != 0)
          encryptBytes(keys, buffer1, read);

        if ( (written = device->write(buffer1, read)) != read )
        {
          actualFile.close();
          delete h;
          return Zip::WriteFailed;
        }
      }
    }
    else
    {
      z_stream zstr;

      // Initialize zalloc, zfree and opaque before calling the init function
      zstr.zalloc = Z_NULL;
      zstr.zfree = Z_NULL;
      zstr.opaque = Z_NULL;

      int zret;

      // Use deflateInit2 with negative windowBits to get raw compression
      if ((zret = deflateInit2_(
          &zstr,
          (int)level,
          Z_DEFLATED,
          -MAX_WBITS,
          8,
          isPNGFile ? Z_RLE : Z_DEFAULT_STRATEGY,
          ZLIB_VERSION,
          sizeof(z_stream)
        )) != Z_OK )
      {
        actualFile.close();
        qDebug() << "Could not initialize zlib for compression";
        delete h;
        return Zip::ZlibError;
      }

      qint64 compressed;

      int flush = Z_NO_FLUSH;

      do
      {
        read = actualFile.read(buffer1, ZIP_READ_BUFFER);
        totRead += read;

        if (read == 0)
          break;
        if (read < 0)
        {
          actualFile.close();
          deflateEnd(&zstr);
          qDebug() << QString("Error while reading %1").arg(file.absoluteFilePath());
          delete h;
          return Zip::ReadFailed;
        }

        crc = crc32(crc, uBuffer, read);

        zstr.next_in = (Bytef*) buffer1;
        zstr.avail_in = (uInt)read;

        // Tell zlib if this is the last chunk we want to encode
        // by setting the flush parameter to Z_FINISH
        flush = (totRead == toRead) ? Z_FINISH : Z_NO_FLUSH;

        // Run deflate() on input until output buffer not full
        // finish compression if all of source has been read in
            do
            {
          zstr.next_out = (Bytef*) buffer2;
          zstr.avail_out = ZIP_READ_BUFFER;

          zret = deflate(&zstr, flush);
          // State not clobbered
          Q_ASSERT(zret != Z_STREAM_ERROR);

          // Write compressed data to file and empty buffer
          compressed = ZIP_READ_BUFFER - zstr.avail_out;

          if (password != 0)
            encryptBytes(keys, buffer2, compressed);

          if (device->write(buffer2, compressed) != compressed)
          {
            deflateEnd(&zstr);
            actualFile.close();
            qDebug() << QString("Error while writing %1").arg(file.absoluteFilePath());
            delete h;
            return Zip::WriteFailed;
          }

          written += compressed;

        } while (zstr.avail_out == 0);

        // All input will be used
        Q_ASSERT(zstr.avail_in == 0);

      } while (flush != Z_FINISH);

      // Stream will be complete
      Q_ASSERT(zret == Z_STREAM_END);

      deflateEnd(&zstr);

    } // if (level != STORE)

    actualFile.close();
  }

  // Store end of entry offset
  quint32 current = device->pos();

  // Update crc and compressed size in local header
  if (!device->seek(crcOffset))
  {
    delete h;
    return Zip::SeekFailed;
  }

  h->crc = dirOnly ? 0 : crc;
  h->szComp += written;

  setULong(h->crc, buffer1, 0);
  setULong(h->szComp, buffer1, 4);
  if ( device->write(buffer1, 8) != 8)
  {
    delete h;
    return Zip::WriteFailed;
  }

  // Seek to end of entry
  if (!device->seek(current))
  {
    delete h;
    return Zip::SeekFailed;
  }

  if ((h->gpFlag[0] & 8) == 8)
  {
    // Write data descriptor

    // Signature: PK\7\8
    buffer1[0] = 'P';
    buffer1[1] = 'K';
    buffer1[2] = 0x07;
    buffer1[3] = 0x08;

    // CRC
    setULong(h->crc, buffer1, ZIP_DD_OFF_CRC32);

    // Compressed size
    setULong(h->szComp, buffer1, ZIP_DD_OFF_CSIZE);

    // Uncompressed size
    setULong(h->szUncomp, buffer1, ZIP_DD_OFF_USIZE);

    if (device->write(buffer1, ZIP_DD_SIZE_WS) !=
ZIP_DD_SIZE_WS)
    {
      delete h;
      return Zip::WriteFailed;
    }
  }

  headers->insert(entryName, h);
  return Zip::Ok;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



4.49.3.4 int decryptByte (quint32 *key2*) const `[inline]`

Definition at line 923 of file zip.cpp.

```
{
  quint16 temp = ((quint16)(key2) & 0xffff) | 2;
  return (int)(((temp * (temp ^ 1)) >> 8) & 0xff);
}
```

4.49.3.5 Zip::CompressionLevel detectCompressionByMime (const QString & *ext*)

Definition at line 979 of file zip.cpp.

```
{
  // files really hard to compress
  if ((ext == "png") ||
    (ext == "jpg") ||
    (ext == "jpeg") ||
    (ext == "mp3") ||
    (ext == "ogg") ||
    (ext == "ogm") ||
    (ext == "avi") ||
    (ext == "mov") ||
    (ext == "rm") ||
    (ext == "ra") ||
    (ext == "zip") ||
    (ext == "rar") ||
    (ext == "bz2") ||
    (ext == "gz") ||
    (ext == "7z") ||
    (ext == "z") ||
    (ext == "jar")
  ) return Zip::Store;

  // files slow and hard to compress
  if ((ext == "exe") ||
    (ext == "bin") ||
    (ext == "rpm") ||
    (ext == "deb")
  ) return Zip::Deflate2;

  return Zip::Deflate9;
}
```

4.49.3.6 void encryptBytes (quint32 ∗ *keys,* char ∗ *buffer,* qint64 *read*) [inline]

Definition at line 966 of file zip.cpp.

```
{
  char t;

  for (int i=0; i<(int)read; ++i)
  {
    t = buffer[i];
    buffer[i] ^= decryptByte(keys[2]);
    updateKeys(keys, t);
  }
}
```

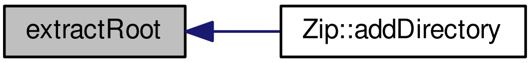
4.49.3.7 QString extractRoot (const QString & *p*) [inline]

Definition at line 1213 of file zip.cpp.

Referenced by Zip::addDirectory().

```
{
  QDir d(QDir::cleanPath(p));
  if (!d.exists())
    return QString();

  if (!d.cdUp())
    return QString();

  return d.absolutePath();
}
```

Here is the caller graph for this function:

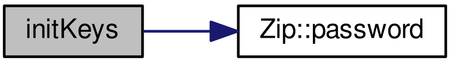


4.49.3.8 void initKeys (quint32 ∗ *keys*) const [inline]

Definition at line 939 of file zip.cpp.

```
{
  // Encryption keys initialization constants are taken from the
  // PKZip file format specification docs
  keys[0] = 305419896L;
  keys[1] = 591751049L;
  keys[2] = 878082192L;

    //QByteArray pwdBytes = password.toAscii();
    QByteArray pwdBytes = password.toLatin1();
  int sz = pwdBytes.size();
  const char* ascii = pwdBytes.data();

  for (int i=0; i<sz; ++i)
    updateKeys(keys, (int)ascii[i]);
}
```

Here is the call graph for this function:



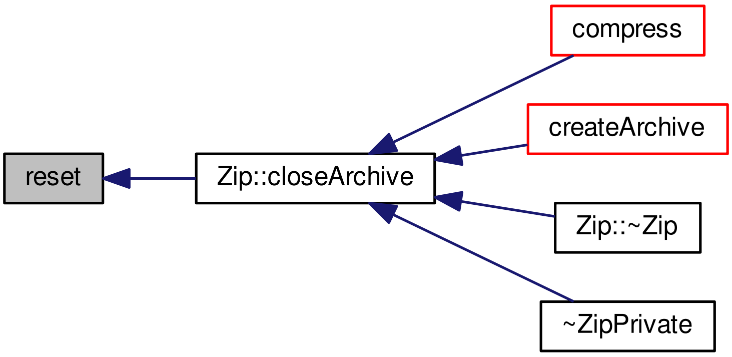
4.49.3.9 void reset ( )

Definition at line 1198 of file zip.cpp.

Referenced by Zip::closeArchive().

```
{
  comment.clear();

  if (headers != 0)
  {
    qDeleteAll(*headers);
    delete headers;
    headers = 0;
  }

  delete device; device = 0;
}
```

Here is the caller graph for this function:



4.49.3.10 void setULong (quint32 *v*, char ∗ *buffer*, unsigned int *offset*) [inline]

Definition at line 930 of file zip.cpp.

```
{
  buffer[offset+3] = ((v >> 24) & 0xFF);
  buffer[offset+2] = ((v >> 16) & 0xFF);
  buffer[offset+1] = ((v >> 8) & 0xFF);
  buffer[offset] = (v & 0xFF);
}
```

4.49.3.11 void updateKeys (quint32 * *keys,* int *c*) const [inline]

Definition at line 957 of file zip.cpp.

```
{
  keys[0] = CRC32(keys[0], c);
  keys[1] += keys[0] & 0xff;
  keys[1] = keys[1] * 134775813L + 1;
  keys[2] = CRC32(keys[2], ((int)keys[1]) >> 24);
}
```

4.49.3.12 bool zLibInit ( )

4.49.4 Member Data Documentation

4.49.4.1 char buffer1[ZIP_READ_BUFFER]

Definition at line 64 of file zip_p.h.

4.49.4.2 char buffer2[ZIP_READ_BUFFER]

Definition at line 65 of file zip_p.h.

4.49.4.3 QString comment

Definition at line 71 of file zip_p.h.

Referenced by Zip::archiveComment(), and Zip::setArchiveComment().

4.49.4.4 const quint32* crcTable

Definition at line 69 of file zip_p.h.

4.49.4.5 QIODevice* device

Definition at line 62 of file zip_p.h.

Referenced by Zip::addDirectory(), Zip::isOpen(), and Zip::setArchiveComment().

4.49.4.6 QMap<QString,ZipEntryP*>* headers

Definition at line 60 of file zip_p.h.

4.49.4.7 QString password

Definition at line 72 of file zip_p.h.

Referenced by Zip::clearPassword(), Zip::password(), and Zip::setPassword().

4.49.4.8 unsigned char∗ uBuffer

Definition at line 67 of file zip_p.h.

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

- /home/lobianco/git/ffsm_pp/src/zip_p.h
- /home/lobianco/git/ffsm_pp/src/zip.cpp

5 File Documentation

5.1 /home/lobianco/git/ffsm_pp/AUTHORS File Reference

5.2 /home/lobianco/git/ffsm_pp/AUTHORS

```
French Forest Sector Model team:
http://ffsm-project.org/wiki/en/team/home#current_team
```

5.3 /home/lobianco/git/ffsm_pp/COPYING File Reference

5.4 /home/lobianco/git/ffsm_pp/COPYING

```
FFSM++ License

This software is covered by the above reported GNU GPL version 3 licence
with the following exceptions that prevail over the GNU GPL version:

#1: Any public communication (not limited: working papers, articles, technical reports)
of results derived from running a modified version of this software requires the
publication of the source code corresponding to such modifications;

#2: Publishing communications derived from unmodified versions of the software
on which new data is applied doesn’t require the publication of the data;

#3: The modifications of which to the first point must be released under the same
licence of the unmodified software, including these exceptions.




                    GNU GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
```

```
free programs, and that you know you can do these things.

  To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights.  Therefore, you have
certain responsibilities if you distribute copies of the software, or if
you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received.  You must make sure that they, too, receive
or can get the source code.  And you must show them these terms so they
know their rights.

  Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License
giving you legal permission to copy, distribute and/or modify it.

  For the developers' and authors' protection, the GPL clearly explains
that there is no warranty for this free software.  For both users' and
authors' sake, the GPL requires that modified versions be marked as
changed, so that their problems will not be attributed erroneously to
authors of previous versions.

  Some devices are designed to deny users access to install or run
modified versions of the software inside them, although the manufacturer
can do so.  This is fundamentally incompatible with the aim of
protecting users' freedom to change the software.  The systematic
pattern of such abuse occurs in the area of products for individuals to
use, which is precisely where it is most unacceptable.  Therefore, we
have designed this version of the GPL to prohibit the practice for those
products.  If such problems arise substantially in other domains, we
stand ready to extend this provision to those domains in future versions
of the GPL, as needed to protect the freedom of users.

  Finally, every program is threatened constantly by software patents.
States should not allow patents to restrict development and use of
software on general-purpose computers, but in those that do, we wish to
avoid the special danger that patents applied to a free program could
make it effectively proprietary.  To prevent this, the GPL assures that
patents cannot be used to render the program non-free.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License.  Each licensee is addressed as "you".  "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy.  The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy.  Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.

  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

```

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users’ Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work’s users, your or third parties’ legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program’s source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any

```
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under section
    7.  This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.

    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source.  This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
    with subsection 6b.

    d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to the
    Corresponding Source in the same way through the same place at no
    further charge.  You need not require recipients to copy the
    Corresponding Source along with the object code.  If the place to
    copy the object code is a network server, the Corresponding Source
    may be on a different server (operated by you or a third party)
    that supports equivalent copying facilities, provided you maintain
    clear directions next to the object code saying where to find the
    Corresponding Source.  Regardless of what server hosts the
    Corresponding Source, you remain obligated to ensure that it is
    available for as long as needed to satisfy these requirements.

    e) Convey the object code using peer-to-peer transmission, provided
    you inform other peers where the object code and Corresponding
    Source of the work are being offered to the general public at no
    charge under subsection 6d.

  A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

  A "User Product" is either (1) a "consumer product", which means any
tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation
into a dwelling.  In determining whether a product is a consumer product,
doubtful cases shall be resolved in favor of coverage.  For a particular
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the status
of the particular user or of the way in which the particular user
actually uses, or expects or is expected to use, the product.  A product
is a consumer product regardless of whether the product has substantial
commercial, industrial or non-consumer uses, unless such uses represent
the only significant mode of use of the product.

  "Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to install
and execute modified versions of a covered work in that User Product from
a modified version of its Corresponding Source.  The information must
suffice to ensure that the continued functioning of the modified object
code is in no case prevented or interfered with solely because
modification has been made.

  If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information.  But this requirement does not apply
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

  The requirement to provide Installation Information does not include a
requirement to continue to provide support service, warranty, or updates
for a work that has been modified or installed by the recipient, or for
the User Product in which it has been modified or installed.  Access to a
network may be denied when the modification itself materially and
adversely affects the operation of the network or violates the rules and
protocols for communication across the network.

  Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.

  7. Additional Terms.

  "Additional permissions" are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law.  If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

  When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it.  (Additional permissions may be written to require their own
removal in certain cases when you modify the work.)  You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders of
that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or

    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or

    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.

  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

  Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License.  If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or
run a copy of the Program.  Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance.  However,
nothing other than this License grants you permission to propagate or
modify any covered work.  These actions infringe copyright if you do
not accept this License.  Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.

  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License.  You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations.  If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.
```

```

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License.  For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based.  The
work thus licensed is called the contributor’s "contributor version".

  A contributor’s "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version.  For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor’s essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement).  To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.

  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients.  "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient’s use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License.  You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  12. No Surrender of Others’ Freedom.

  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all.  For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
```

```
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

  13. Use with the GNU Affero General Public License.

  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a single
combined work, and to convey the resulting work.  The terms of this
License will continue to apply to the part which is the covered work,
but the special requirements of the GNU Affero General Public License,
section 13, concerning interaction through a network will apply to the
combination as such.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

  Each version is given a distinguishing version number.  If the
Program specifies that a certain numbered version of the GNU General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation.  If the Program does not specify a version number of the
GNU General Public License, you may choose any version ever published
by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy’s
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

  Later license versions may give you additional or different
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program’s name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate
parts of the General Public License.  Of course, your program’s commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<http://www.gnu.org/philosophy/why-not-lgpl.html>.
```

5.5 /home/lobianco/git/ffsm_pp/data/gis/france/clc_311.grd.aux.xml File Reference

5.6 clc_311.grd.aux.xml

```xml
<PAMDataset>
  <PAMRasterBand band="1">
    <Histograms>
      <HistItem>
        <HistMin>-37646.3995</HistMin>
        <HistMax>55312447.3995</HistMax>
        <BucketCount>1000</BucketCount>
        <IncludeOutOfRange>0</IncludeOutOfRange>
        <Approximate>0</Approximate>
        <HistCounts>19997|54|43|35|34|78|63|55|50|44|48|51|53|41|52|58|51|59|40|48|37|45|45|37|28|43|28|36|
28|40|35|33|34|38|30|43|28|34|34|36|33|30|33|31|32|37|36|15|28|32|33|34|35|27|26|22|27|24|24|37|32|28|29|16|
21|29|21|28|29|18|31|27|32|24|31|15|23|34|27|20|27|25|15|34|20|16|26|31|20|22|31|27|29|18|24|22|22|29|21|23|
20|19|21|23|27|37|20|24|18|20|20|18|21|20|18|18|25|22|24|19|27|22|22|20|20|21|20|23|20|11|14|24|18|23|21|24|
15|19|22|14|18|17|12|14|21|18|15|15|23|17|15|14|21|17|13|15|19|19|26|26|14|25|19|20|16|12|18|10|18|14|17|11|
14|11|13|17|26|14|16|17|18|14|8|16|14|13|8|13|15|9|9|10|19|14|15|14|11|15|15|18|20|17|11|11|10|14|16|17|12|2
1|19|18|17|18|9|20|27|10|10|9|11|17|9|12|12|10|19|7|11|10|17|14|13|10|13|14|16|16|14|8|12|12|13|14|9|14|10|9
|11|10|11|17|5|8|13|11|14|5|14|13|8|10|13|6|6|14|12|6|13|11|10|12|10|16|8|15|8|12|9|11|9|10|13|9|10|8|9|9|9|
10|12|6|11|16|7|9|17|7|11|11|14|11|9|12|9|11|13|13|9|9|11|10|8|11|10|7|12|11|14|9|10|11|9|11|6|10|5|10|6|10|
10|8|11|5|8|7|14|12|8|6|7|9|7|10|6|4|9|11|13|8|12|12|7|8|5|9|7|11|8|5|6|9|12|7|8|2|12|7|10|8|7|2|8|4|8|7|7|5
|6|9|12|5|6|6|7|6|5|7|6|8|6|3|11|6|6|4|4|4|5|7|7|9|11|7|6|11|6|5|8|2|6|8|7|8|8|9|7|9|9|2|2|4|7|8|12|4|3|7|6|
10|9|13|6|3|4|6|9|7|8|7|4|4|4|3|6|6|1|6|7|9|4|6|5|4|4|5|6|5|5|6|5|4|2|11|6|5|10|3|6|6|4|5|3|9|7|7|9|5|5|6|3|
4|6|5|4|5|5|6|7|5|4|4|5|6|5|7|5|1|7|5|3|3|5|5|2|2|6|3|8|2|7|2|1|2|3|2|4|5|3|0|8|5|3|2|3|7|5|1|5|7|6|2|5|4|6|
3|2|3|6|4|3|3|2|3|0|3|3|3|3|2|3|6|1|5|1|4|2|4|5|0|3|2|2|0|4|5|4|1|1|2|2|4|1|4|0|3|5|1|7|0|2|2|4|1|4|2|2|0|3|
3|1|2|3|5|2|4|1|0|5|1|2|2|1|2|4|1|3|1|4|2|0|2|4|0|1|2|1|1|1|2|1|4|0|1|2|1|1|2|2|3|4|1|1|0|1|1|0|2|2|3|2|3|0|
1|4|2|1|0|5|3|1|2|4|2|1|3|3|1|1|0|0|0|1|1|1|2|2|1|1|1|0|1|2|0|1|1|1|1|1|1|1|1|0|1|1|0|0|1|1|1|1|1|2|1|1|0|2|
1|0|1|1|0|1|2|1|0|0|1|2|0|3|0|0|3|2|0|1|2|1|2|1|0|0|0|1|1|4|2|2|2|0|2|1|1|1|0|1|2|0|1|1|1|0|0|2|1|0|0|1|3|0|
1|1|0|1|0|1|0|0|1|1|2|0|0|0|1|0|0|0|0|0|0|1|2|0|0|0|1|0|0|2|2|0|1|0|0|0|1|0|0|0|0|0|0|1|1|0|0|0|0|1|0|2|0|
0|0|0|1|1|0|0|0|0|0|0|1|2|0|0|1|1|0|0|0|1|0|0|1|0|0|0|0|0|0|0|0|0|1|1|0|0|0|0|1|0|0|1|1|0|0|0|0|1|0|0|0|0|0|
0|0|0|0|0|0|0|1|0|0|0|0|1|0|0|0|1|1|0|0|0|1|0|0|0|0|1|0|0|0|1|0|0|0|1|0|0|0|0|0|1|0|1|0|0|0|0|0|0|0|0|2|0|0|
0|0|0|1|1|0|0|0|1|0|0|0|0|0|0|1|0|1|1|0|1|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|
0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|2|1</HistCounts>
      </HistItem>
    </Histograms>
    <Metadata>
      <MDI key="STATISTICS_MAXIMUM">55284800</MDI>
      <MDI key="STATISTICS_MEAN">3108511.4642149</MDI>
      <MDI key="STATISTICS_MINIMUM">-9999</MDI>
      <MDI key="STATISTICS_STDDEV">7146798.9379336</MDI>
    </Metadata>
  </PAMRasterBand>
</PAMDataset>
```

5.7 /home/lobianco/git/ffsm_pp/data/gis/france/nut_l2.grd.aux.xml File Reference

5.8 nut_l2.grd.aux.xml

```
<PAMDataset>
  <PAMRasterBand band="1">
    <Histograms>
      <HistItem>
        <HistMin>-10009.541</HistMin>
        <HistMax>11093.541</HistMax>
        <BucketCount>1000</BucketCount>
        <IncludeOutOfRange>0</IncludeOutOfRange>
        <Approximate>0</Approximate>
        <HistCounts>19644|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0
|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|0|2675|1253|3588|1064</HistCounts>
      </HistItem>
    </Histograms>
    <Metadata>
      <MDI key="STATISTICS_MAXIMUM">11083</MDI>
      <MDI key="STATISTICS_MEAN">-3600.121704932</MDI>
      <MDI key="STATISTICS_MINIMUM">-9999</MDI>
      <MDI key="STATISTICS_STDDEV">9682.231972089</MDI>
    </Metadata>
  </PAMRasterBand>
</PAMDataset>
```

5.9 /home/lobianco/git/ffsm_pp/data/gis/gis_france.xml File Reference

5.10 gis_france.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<gis>
  <layer>
    <!-- Keeps the names as forType_FORNAME -->
    <name>forArea_broadL</name>
    <label>Broad-leaved forest areas</label>
    <comment>Derived from clc06, theme311. The area of Broad-leaved forest areas within each cell.</comment
>
    <isInteger>false</isInteger><!-- Bool. If "isInteger" is not true, minValue and maxValue must be
 specified for each legendItem -->
    <readAtStart>true</readAtStart><!-- bool -->
    <dynamicContent>true</dynamicContent><!-- bool. True if this layer may change during simulationj
 period, false if it is fixed -->
    <expandByFt>false</expandByFt><!-- bool. If true this ft expand for each forest type -->
    <dirName>gis/france/</dirName>
    <fileName>clc_311.grd</fileName>
    <legendItems>
      <legendItem label="min10%" rColor="240" gColor="255" bColor="240" minValue="0" maxValue="8000000">00
</legendItem>
      <legendItem label="10-20%" rColor="245" gColor="245" bColor="245" minValue="8000000" maxValue=
"16000000">10</legendItem>
```

```
      <legendItem label="20-30%" rColor="205" gColor="225" bColor="205" minValue="16000000" maxValue="24000000">20</legendItem>
      <legendItem label="30-40%" rColor="185" gColor="205" bColor="185" minValue="24000000" maxValue="32000000">30</legendItem>
      <legendItem label="40-50%" rColor="160" gColor="180" bColor="160" minValue="32000000" maxValue="40000000">40</legendItem>
      <legendItem label="50-60%" rColor="135" gColor="155" bColor="135" minValue="40000000" maxValue="48000000">50</legendItem>
      <legendItem label="60-70%" rColor="110" gColor="130" bColor="110" minValue="48000000" maxValue="56000000">60</legendItem>
      <legendItem label="70-80%" rColor="085" gColor="105" bColor="085" minValue="56000000" maxValue="64000000">70</legendItem>
      <legendItem label="80-90%" rColor="060" gColor="080" bColor="060" minValue="64000000" maxValue="72000000">80</legendItem>
      <legendItem label="90-100" rColor="035" gColor="055" bColor="035" minValue="72000000" maxValue="80000000">90</legendItem>
    </legendItems>
  </layer>
  <layer>
    <name>forArea_con</name>
    <label>Coniferous forest areas</label>
    <comment>Derived from clc06, theme312. The area of Broad-leaved forest areas within each cell.</comment>
    <isInteger>false</isInteger><!-- Bool. If "isInteger" is not true, minValue and maxValue must be specified for each legendItem -->
    <readAtStart>true</readAtStart><!-- bool -->
    <dynamicContent>true</dynamicContent><!-- bool. True if this layer may change during simulationj period, false if it is fixed -->
    <expandByFt>false</expandByFt><!-- bool. If true this ft expand for each forest type -->
    <dirName>gis/france/</dirName>
    <fileName>clc_312.grd</fileName>
    <legendItems>
      <legendItem label="min10%" rColor="240" gColor="255" bColor="240" minValue="0" maxValue="8000000">00</legendItem>
      <legendItem label="10-20%" rColor="245" gColor="245" bColor="245" minValue="8000000" maxValue="16000000">10</legendItem>
      <legendItem label="20-30%" rColor="205" gColor="225" bColor="205" minValue="16000000" maxValue="24000000">20</legendItem>
      <legendItem label="30-40%" rColor="185" gColor="205" bColor="185" minValue="24000000" maxValue="32000000">30</legendItem>
      <legendItem label="40-50%" rColor="160" gColor="180" bColor="160" minValue="32000000" maxValue="40000000">40</legendItem>
      <legendItem label="50-60%" rColor="135" gColor="155" bColor="135" minValue="40000000" maxValue="48000000">50</legendItem>
      <legendItem label="60-70%" rColor="110" gColor="130" bColor="110" minValue="48000000" maxValue="56000000">60</legendItem>
      <legendItem label="70-80%" rColor="085" gColor="105" bColor="085" minValue="56000000" maxValue="64000000">70</legendItem>
      <legendItem label="80-90%" rColor="060" gColor="080" bColor="060" minValue="64000000" maxValue="72000000">80</legendItem>
      <legendItem label="90-100" rColor="035" gColor="055" bColor="035" minValue="72000000" maxValue="80000000">90</legendItem>
    </legendItems>

  </layer>
  <layer>
    <name>forArea_mixedBC</name>
    <label>Mixed forest forest areas</label>
    <comment>Derived from clc06, theme313. The area of Mixed Broad-leaved/Coniferous forest areas within each cell.</comment>
    <isInteger>false</isInteger><!-- Bool. If "isInteger" is not true, minValue and maxValue must be specified for each legendItem -->
    <readAtStart>true</readAtStart><!-- bool -->
    <dynamicContent>true</dynamicContent><!-- bool. True if this layer may change during simulationj period, false if it is fixed -->
    <expandByFt>false</expandByFt><!-- bool. If true this ft expand for each forest type -->
    <dirName>gis/france/</dirName>
    <fileName>clc_313.grd</fileName>
    <legendItems>
      <legendItem label="min10%" rColor="240" gColor="255" bColor="240" minValue="0" maxValue="8000000">00</legendItem>
      <legendItem label="10-20%" rColor="245" gColor="245" bColor="245" minValue="8000000" maxValue="16000000">10</legendItem>
      <legendItem label="20-30%" rColor="205" gColor="225" bColor="205" minValue="16000000" maxValue="24000000">20</legendItem>
      <legendItem label="30-40%" rColor="185" gColor="205" bColor="185" minValue="24000000" maxValue="32000000">30</legendItem>
      <legendItem label="40-50%" rColor="160" gColor="180" bColor="160" minValue="32000000" maxValue="40000000">40</legendItem>
      <legendItem label="50-60%" rColor="135" gColor="155" bColor="135" minValue="40000000" maxValue="48000000">50</legendItem>
      <legendItem label="60-70%" rColor="110" gColor="130" bColor="110" minValue="48000000" maxValue="56000000">60</legendItem>
      <legendItem label="70-80%" rColor="085" gColor="105" bColor="085" minValue="56000000" maxValue="64000000">70</legendItem>
      <legendItem label="80-90%" rColor="060" gColor="080" bColor="060" minValue="64000000" maxValue="72000000">80</legendItem>
      <legendItem label="90-100" rColor="035" gColor="055" bColor="035" minValue="72000000" maxValue=
"80000000">90</legendItem>
    </legendItems>
  </layer>
    <layer>
    <name>forArea_Oth</name>
    <label>Other, non forest areas</label>
    <comment>Derived from clc06. The area of NON forest areas within each cell.</comment>
    <isInteger>false</isInteger><!-- Bool. If "isInteger" is not true, minValue and maxValue must be
 specified for each legendItem -->
    <readAtStart>false</readAtStart><!-- bool -->
    <dynamicContent>false</dynamicContent><!-- bool. True if this layer may change during simulationj
 period, false if it is fixed -->
    <expandByFt>false</expandByFt><!-- bool. If true this ft expand for each forest type -->
    <dirName>gis/france/</dirName>
    <fileName>clc_999.grd</fileName>
    <legendItems>
      <legendItem label="min10%" rColor="255" gColor="255" bColor="255" minValue="0" maxValue="8000000">00
</legendItem>
      <legendItem label="10-20%" rColor="240" gColor="240" bColor="240" minValue="8000000" maxValue=
"16000000">10</legendItem>
      <legendItem label="20-30%" rColor="220" gColor="220" bColor="220" minValue="16000000" maxValue=
"24000000">20</legendItem>
      <legendItem label="30-40%" rColor="200" gColor="200" bColor="200" minValue="24000000" maxValue=
"32000000">30</legendItem>
      <legendItem label="40-50%" rColor="175" gColor="175" bColor="175" minValue="32000000" maxValue=
"40000000">40</legendItem>
      <legendItem label="50-60%" rColor="150" gColor="150" bColor="150" minValue="40000000" maxValue=
"48000000">50</legendItem>
      <legendItem label="60-70%" rColor="125" gColor="125" bColor="125" minValue="48000000" maxValue=
"56000000">60</legendItem>
      <legendItem label="70-80%" rColor="100" gColor="100" bColor="100" minValue="56000000" maxValue=
"64000000">70</legendItem>
      <legendItem label="80-90%" rColor="075" gColor="075" bColor="075" minValue="64000000" maxValue=
"72000000">80</legendItem>
      <legendItem label="90-100" rColor="050" gColor="050" bColor="050" minValue="72000000" maxValue=
"80000000">90</legendItem>
    </legendItems>
    <reclassRules/>
  </layer>
  <layer>
    <!-- Keeps the names as regLev_LEVELNUMBER -->
    <name>regLev_2</name>
    <label>Subregions (nuts2)</label>
    <isInteger>true</isInteger><!-- Bool. If "isInteger" is not true, minValue and maxValue must be
 specified for each legendItem -->
    <readAtStart>true</readAtStart><!-- bool -->
    <dynamicContent>false</dynamicContent><!-- bool. True if this layer may change during simulationj
 period, false if it is fixed -->
    <expandByFt>false</expandByFt><!-- bool. If true this ft expand for each forest type -->
    <dirName>gis/france/</dirName>
    <fileName>nut_l2.grd</fileName>
    <legendItems>
      <legendItem label="AL - Alsace" rColor="230" gColor="010" bColor="077">11042</legendItem>
      <legendItem label="AQ - Aquitaine" rColor="230" gColor="020" bColor="077">11061</legendItem>
      <legendItem label="AU - Auvergne" rColor="230" gColor="030" bColor="077">11072</legendItem>
      <legendItem label="BN - Basse-Normandie" rColor="230" gColor="040" bColor="077">11025</legendItem>
      <legendItem label="BO - Bourgogne" rColor="230" gColor="050" bColor="077">11026</legendItem>
      <legendItem label="BR - Bretagne" rColor="230" gColor="060" bColor="077">11052</legendItem>
      <legendItem label="CE - Centre" rColor="230" gColor="070" bColor="077">11024</legendItem>
      <legendItem label="CA - Champagne-Ardennes" rColor="230" gColor="080" bColor="077">11021</legendItem>
      <legendItem label="CO - Corse" rColor="230" gColor="090" bColor="077">11083</legendItem>
      <legendItem label="FC - Franche-Compté" rColor="230" gColor="100" bColor="077">11043</legendItem>
      <legendItem label="HN - Haute-Normandie" rColor="230" gColor="110" bColor="077">11023</legendItem>
      <legendItem label="IF - Ile-de-France" rColor="230" gColor="120" bColor="077">11010</legendItem>
      <legendItem label="LR - Languedoc-Roussillon" rColor="230" gColor="130" bColor="077">11081</
legendItem>
      <legendItem label="LI - Limousin" rColor="230" gColor="140" bColor="077">11063</legendItem>
      <legendItem label="LO - Lorraine" rColor="230" gColor="150" bColor="077">11041</legendItem>
      <legendItem label="MP - Midi-Pyrénées" rColor="230" gColor="160" bColor="077">11062</legendItem>
      <legendItem label="NP - Nord-Pas-de-Calais" rColor="230" gColor="170" bColor="077">11030</legendItem>
      <legendItem label="PL - Pays de la Loire" rColor="230" gColor="180" bColor="077">11051</legendItem>
      <legendItem label="PI - Picardie" rColor="230" gColor="190" bColor="077">11022</legendItem>
      <legendItem label="PC - Poitou-Charentes" rColor="230" gColor="200" bColor="077">11053</legendItem>
      <legendItem label="PA - Provence-Alpes-Côte-d’Azur" rColor="230" gColor="210" bColor="077">11082</
legendItem>
      <legendItem label="RA - Rhône-Alpes" rColor="230" gColor="220" bColor="077">11071</legendItem>
    </legendItems>
  </layer>
    <layer>
    <name>regLev_1</name>
    <label>Regions (counties, nuts1)</label>
    <isInteger>true</isInteger><!-- Bool. If "isInteger" is not true, minValue and maxValue must be
 specified for each legendItem -->
    <readAtStart>true</readAtStart><!-- bool -->
    <dynamicContent>false</dynamicContent><!-- bool. True if this layer may change during simulationj
 period, false if it is fixed -->
```

```
    <expandByFt>false</expandByFt><!-- bool. If true this ft expand for each forest type -->
    <dirName>gis/france/</dirName>
    <fileName>nut_l1.grd</fileName>
    <legendItems>
      <legendItem label="FR - France" rColor="130" gColor="70" bColor="180">11000</legendItem>
    </legendItems>
  </layer>
  <layer>
    <name>dtm</name>
    <label>Digital Terrain Model</label>
    <isInteger>false</isInteger><!-- Bool. If "isInteger" is not true, minValue and maxValue must be specified for each legendItem -->
    <readAtStart>true</readAtStart><!-- bool -->
    <dynamicContent>false</dynamicContent><!-- bool. True if this layer may change during simulationj period, false if it is fixed -->
    <expandByFt>false</expandByFt><!-- bool. If true this ft expand for each forest type -->
    <dirName>gis/france/</dirName>
    <fileName>dtm_8000m.grd</fileName>
    <legendItems>
      <legendItem label="0 - 100 m" rColor="255" gColor="255" bColor="255" minValue="0" maxValue="100">0</legendItem>
      <legendItem label="100 - 200 m" rColor="240" gColor="240" bColor="240" minValue="100" maxValue="200">100</legendItem>
      <legendItem label="200 - 300 m" rColor="220" gColor="220" bColor="220" minValue="200" maxValue="300">200</legendItem>
      <legendItem label="300 - 400 m" rColor="200" gColor="200" bColor="200" minValue="300" maxValue="400">300</legendItem>
      <legendItem label="400 - 500 m" rColor="175" gColor="175" bColor="175" minValue="400" maxValue="500">400</legendItem>
      <legendItem label="500 - 600 m" rColor="150" gColor="150" bColor="150" minValue="500" maxValue="600">500</legendItem>
      <legendItem label="600 - 700 m" rColor="125" gColor="125" bColor="125" minValue="600" maxValue="700">600</legendItem>
      <legendItem label="700 - 800 m" rColor="100" gColor="100" bColor="100" minValue="700" maxValue="800">700</legendItem>
      <legendItem label="800 - 900 m" rColor="075" gColor="075" bColor="075" minValue="800" maxValue="900">800</legendItem>
      <legendItem label="900 - 1000 m" rColor="050" gColor="050" bColor="050" minValue="900" maxValue="1000">900</legendItem>
      <legendItem label="over 1000 m" rColor="025" gColor="025" bColor="025" minValue="1000" maxValue="10000">1000</legendItem>
    </legendItems>
  </layer>
  <layer>
    <name>vol</name>
    <label>Volume</label>
    <isInteger>false</isInteger><!-- Bool. If "isInteger" is not true, minValue and maxValue must be specified for each legendItem -->
    <readAtStart>true</readAtStart><!-- bool -->
    <dynamicContent>true</dynamicContent><!-- bool. True if this layer may change during simulationj period, false if it is fixed -->
    <expandByFt>true</expandByFt><!-- bool. If true this ft expand for each forest type -->
    <dirName></dirName>
    <fileName></fileName>
    <legendItems>
      <!-- id must be an integer. For float layers is not used any how other than to check the legend item already exists. -->
      <legendItem label="0     - 0.001 Mmc" rColor="255" gColor="255" bColor="255" minValue="0" maxValue="0.001">1</legendItem>
      <legendItem label="0.001 - 0.005 Mmc" rColor="240" gColor="240" bColor="240" minValue="0.001" maxValue="0.005">2</legendItem>
      <legendItem label="0.005 - 0.01  Mmc" rColor="220" gColor="220" bColor="220" minValue="0.005" maxValue="0.01">3</legendItem>
      <legendItem label="0.01  - 0.05  Mmc" rColor="200" gColor="200" bColor="200" minValue="0.01" maxValue="0.05">4</legendItem>
      <legendItem label="0.05  - 0.1   Mmc" rColor="175" gColor="175" bColor="175" minValue="0.05" maxValue="0.1">5</legendItem>
      <legendItem label="0.1   - 0.2   Mmc" rColor="150" gColor="150" bColor="150" minValue="0.1" maxValue="0.2">6</legendItem>
      <legendItem label="0.2   - 0.5   Mmc" rColor="125" gColor="125" bColor="125" minValue="0.2" maxValue="0.5">7</legendItem>
      <legendItem label="0.5   - 1     Mmc" rColor="100" gColor="100" bColor="100" minValue="0.5" maxValue="1">8</legendItem>
      <legendItem label="1     - 1.5   Mmc" rColor="075" gColor="075" bColor="075" minValue="1" maxValue="1.5">9</legendItem>
      <legendItem label="1.5   - 2     Mmc" rColor="050" gColor="050" bColor="050" minValue="1.5" maxValue="2">10</legendItem>
      <legendItem label="over 2 Mmc"        rColor="025" gColor="025" bColor="025" minValue="2" maxValue="10000">11</legendItem>
    </legendItems>
  </layer>
  <layer>
    <name>expectedReturns</name>
    <label>Expected returns</label>
    <isInteger>false</isInteger><!-- Bool. If "isInteger" is not true, minValue and maxValue must be specified for each legendItem -->
    <readAtStart>true</readAtStart><!-- bool -->
    <dynamicContent>true</dynamicContent><!-- bool. True if this layer may change during simulationj
 period, false if it is fixed -->
    <expandByFt>true</expandByFt><!-- bool. If true this ft expand for each forest type -->
    <dirName></dirName>
    <fileName></fileName>
    <legendItems>
      <legendItem label="0 - 5 e/ha" rColor="255" gColor="255" bColor="255" minValue="0" maxValue="5">0</
legendItem>
      <legendItem label="5 - 20 e/ha" rColor="240" gColor="240" bColor="240" minValue="5" maxValue="20">5</
legendItem>
      <legendItem label="20 - 50 e/ha" rColor="220" gColor="220" bColor="220" minValue="20" maxValue="50">2
0</legendItem>
      <legendItem label="50 - 100 e/ha" rColor="200" gColor="200" bColor="200" minValue="50" maxValue="100"
>50</legendItem>
      <legendItem label="100 - 200 e/ha" rColor="175" gColor="175" bColor="175" minValue="100" maxValue=
"200">100</legendItem>
      <legendItem label="200 - 500 e/ha" rColor="150" gColor="150" bColor="150" minValue="200" maxValue=
"500">200</legendItem>
      <legendItem label="500 - 1000 e/ha" rColor="125" gColor="125" bColor="125" minValue="500" maxValue=
"1000">500</legendItem>
      <legendItem label="1000 - 2000 e/ha" rColor="100" gColor="100" bColor="100" minValue="1000" maxValue=
"2000">1000</legendItem>
      <legendItem label="2000 - 5000 e/ha" rColor="075" gColor="075" bColor="075" minValue="2000" maxValue=
"5000">2000</legendItem>
      <legendItem label="5000 - 10000 e/ha" rColor="050" gColor="050" bColor="050" minValue="5000" maxValue
="10000">5000</legendItem>
      <legendItem label="over 10000 e/ha" rColor="025" gColor="025" bColor="025" minValue="10000" maxValue=
"100000000">10000</legendItem>
    </legendItems>
  </layer>
  <layer>
    <name>avalCoef</name>
    <label>Availability coefficient</label>
    <isInteger>false</isInteger><!-- Bool. If "isInteger" is not true, minValue and maxValue must be
 specified for each legendItem -->
    <readAtStart>true</readAtStart><!-- bool -->
    <dynamicContent>false</dynamicContent><!-- bool. True if this layer may change during simulationj
 period, false if it is fixed -->
    <expandByFt>false</expandByFt><!-- bool. If true this ft expand for each forest type -->
    <dirName>gis/france/</dirName>
    <fileName>avalcoef.grd</fileName>
    <legendItems>
      <legendItem label="0 - 0.2" rColor="230" gColor="230" bColor="230" minValue="0" maxValue="0.2">0</
legendItem>
      <legendItem label="0.2 - 0.4" rColor="160" gColor="160" bColor="160" minValue="0.2" maxValue="0.4">20
</legendItem>
      <legendItem label="0.4 - 0-6" rColor="100" gColor="100" bColor="100" minValue="0.4" maxValue="0.6">40
</legendItem>
      <legendItem label="0.8 - 1" rColor="40" gColor="40" bColor="40" minValue="0.8" maxValue="1.001">80</
legendItem>
    </legendItems>
  </layer>


</gis>



```

5.11 /home/lobianco/git/ffsm_pp/data/output/clean.sh File Reference

5.12 clean.sh

```
#!/bin/bash

#---------------------------------------
#  Shell script to clean the FFSM++ output
#---------------------------------------

echo Cleaning the FFSM++ output...
echo ""

# maps...
rm -rf maps/asciiGrids/*
rm -rf maps/bitmaps/*
rm -rf maps/cats/*
rm -rf maps/colr/*
rm -rf maps/grass/france/default
rm -rf maps/floatListLayers/*
rm -rf maps/integerListLayers/*
rm -rf maps/scenarioNames/*
```

```
# results...
rm -rf results/*.csv
# charts..
rm -rf charts/*.pdf
rm -rf charts/png/*.png
# tables..
rm -rf tables/*
# optimisation logs
rm -rf optimisationLogs/*
# debugs..
rm -rf debugs/*

# copy back the do-not-remove warning file..
cp 00_doNotRemove.txt maps/asciiGrids/
cp 00_doNotRemove.txt maps/bitmaps/
cp 00_doNotRemove.txt maps/cats/
cp 00_doNotRemove.txt maps/colr/
cp 00_doNotRemove.txt maps/floatListLayers/
cp 00_doNotRemove.txt maps/integerListLayers/
cp 00_doNotRemove.txt maps/scenarioNames/
cp 00_doNotRemove.txt charts/
cp 00_doNotRemove.txt charts/png/
cp 00_doNotRemove.txt tables/
cp 00_doNotRemove.txt optimisationLogs/
cp 00_doNotRemove.txt debugs/

# cp the results ods template
cp results_template.ods results/results.ods

echo Done cleaning FFSM++ output!
echo ""
```

5.13 /home/lobianco/git/ffsm_pp/data/output/merge_example.py File Reference

Namespaces

- merge_example

Variables

- list forIFiles
- list prdIFiles
- list carbonIFiles
- list scenarios
- string forOFilename = 'results/forestData_merged.csv'
- string prdOFilename = 'results/productData_merged.csv'
- string carbonOFilename = 'results/carbonBalance_merged.csv'

5.14 merge_example.py

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from merge_lib import *

forIFiles = [
    'results/forestData.csv',
]
prdIFiles = [
    'results/productData.csv',
]
carbonIFiles = [
    'results/carbonBalance.csv',
]
scenarios = [
  'test',
  'test2',
]

forOFilename = 'results/forestData_merged.csv'
prdOFilename = 'results/productData_merged.csv'
carbonOFilename = 'results/carbonBalance_merged.csv'

merge(forIFiles,prdIFiles,carbonIFiles,scenarios,forOFilename,prdOFilename,carbonOFilename)
```

5.15 /home/lobianco/git/ffsm_pp/data/output/merge_lib.py File Reference

Namespaces

- merge_lib

Functions

- def merge (forIFiles_h=[ ], prdIFiles_h=[ ], carbonIFiles_h=[ ], scenarios_h=[ ], forOFilename_h="", prdOFilename_h="", carbonOFilename_h="", variables_h=[ ], regions_h=[ ], years_h=[ ])
- def determinePositions (headerRow)
- def merge_single_file (i_filename_h, o_filename_h, scenarios_h, keepHeader=False, variables_h=[ ], regions_h=[ ], years_h=[ ])

5.16 merge_lib.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# =============================================================================
def merge(
forIFiles_h=[],prdIFiles_h=[],carbonIFiles_h=[],scenarios_h=[],forOFilename_h="",prdOFilename_h="",carbonOFilename_h="",
  print("*** Processing..")
  if len(forIFiles_h)>0:
   open(forOFilename_h,’w’).close()
  if len(prdIFiles_h)>0:
    open(prdOFilename_h,’w’).close()
  if len(carbonIFiles_h)>0:
    open(carbonOFilename_h,’w’).close()
  forCounter=0
  prdCounter=0
  carbonCounter=0
  for forIFile in forIFiles_h:
    merge_single_file(forIFile, forOFilename_h, scenarios_h, False if forCounter else True
, variables_h, regions_h, years_h)
    forCounter += 1
  for prdIFile in prdIFiles_h:
    merge_single_file(prdIFile, prdOFilename_h, scenarios_h, False if prdCounter else True
, variables_h, regions_h, years_h)
    prdCounter += 1
  for carbonIFile in carbonIFiles_h:
    merge_single_file(carbonIFile, carbonOFilename_h, scenarios_h, False if carbonCounter
else True, variables_h, regions_h, years_h)
    carbonCounter += 1
  print ("*** Done!")


# =============================================================================
def determinePositions(headerRow):
  fields = headerRow.split(’;’)
  returnValues = [-1,-1,-1]
  for idx, field in enumerate(fields):
    if(field == ’parName’ or field == ’balItem’): returnValues[0] = idx
    if(field == ’region’):                        returnValues[1] = idx
    if(field == ’year’):                          returnValues[2] = idx
  if (returnValues[0] == -1 or returnValues[1] == -1 or returnValues[2] == -1):
    print ("There has been an error reading the headers of a file.")
    exit(1)
  return returnValues

# =============================================================================
def merge_single_file(i_filename_h, o_filename_h, scenarios_h, keepHeader=False,
variables_h=[], regions_h=[], years_h=[]):
  i_file = open(i_filename_h,’r’)
  o_file = open(o_filename_h,’a’)
  newRow     =  1
  counterRow =  0
  parNamePos = -1
  regionPos  = -1
  yearPos    = -1
  positions  = []

  while newRow:
    row = i_file.readline()
    scenarioFilter = False
    variableFilter = False
    regionFilter   = False
    yearFilter     = False
    finalFilter    = False

    if row == ’’:
      break
    if(counterRow == 0):
      positions = determinePositions(row)
      parNamePos = positions[0]
      regionPos  = positions[1]
      yearPos    = positions[2]
      if(keepHeader):
        o_file.write(row)
    counterRow += 1
    fields = row.split(’;’)
    rowScenario = fields[0]

    if(rowScenario in scenarios_h):
      scenarioFilter = True

    if( (len(variables_h) == 0 ) or (fields[parNamePos] in variables_h) ):
      variableFilter = True

    if( (len(regions_h) == 0) or (fields[regionPos] in regions_h) ):
      regionFilter = True

    if( (len(years_h) == 0) or (fields[yearPos] in years_h) ):
      yearFilter = True

    if (scenarioFilter and variableFilter and regionFilter and yearFilter):
      finalFilter = True

    if(finalFilter):
      o_file.write(row)
  i_file.close()
  o_file.close()

```

5.17 /home/lobianco/git/ffsm_pp/data/output/output_parser_example.py File Reference

Namespaces

- output_parser_example

Functions

- def main ()
- def override_globals ()
- def printCharts ()
- def printTables ()
- def printAATables ()

5.18 output_parser_example.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import os, sys
import csv, math
import matplotlib.pyplot as plt
import output_parser_globals as g
from output_parser_lib import *


# =============================================================================
def main():

  override_globals()
  prepare_data()
  reset_output()

  # H - Printing charts
  if g.printChartsFlag:
    printCharts()

  # I - Print tables
  if g.printTablesFlag:
    printTables()

  # L - Print area allocation confrontation
  if g.printAATablesFlag:
    printAATables()

  print "Done!"

# =============================================================================
def override_globals():

  g.forIFiles = [
    ’results/forestData_baseline.csv’,
    ’results/forestData_constant.csv’,
    ’results/forestData_Ph_L.csv’,
    ’results/forestData_Ph_U.csv’,
    ’results/forestData_Pr_C.csv’,
    ’results/forestData_Pr_U.csv’,
    ’results/forestData_Exp_0.csv’,
    ’results/forestData_Exp_1.csv’,
    ’results/forestData_EOL_en_U.csv’,
  ]

  g.carbonIFiles = [
    ’results/carbonBalance_baseline.csv’,
    ’results/carbonBalance_constant.csv’,
    ’results/carbonBalance_Ph_L.csv’,
    ’results/carbonBalance_Ph_U.csv’,
    ’results/carbonBalance_Pr_C.csv’,
    ’results/carbonBalance_Pr_U.csv’,
    ’results/carbonBalance_Exp_0.csv’,
    ’results/carbonBalance_Exp_1.csv’,
    ’results/carbonBalance_EOL_en_U.csv’,
  ]

  g.scenarios = {
    ’baseline’:             ’#000000’,  # Black
    ’constant’:             ’#cccccc’,  # Grey
    ’Ph_L’:                 ’#b5ff95’,  # Light green
    ’Ph_U’:                 ’#f40303’,  # Red
    ’Pr_C’:                 ’#b5ff95’,  # Light green
    ’Pr_U’:                 ’#f40303’,  # Red
    ’Exp_0’:                ’#b5ff95’,  # Light green
    ’Exp_1’:                ’#f40303’,  # Red
    ’EOL_en_U’:             ’#011bb7’,  # Ink blue

  }
  g.years = [str(y) for y in range(2007,2101)]      # [2007-2100]
  g.printChartsFlag = True
  g.printTablesFlag = True
  g.printAATablesFlag = False
  g.chartoutdir = ’charts’
  g.tableoutdir = ’tables’
  # key: var short name
  # value: turple with long name, unit and optionally variable to act for ponderation and name of aggregated variable
  g.forVars = {’hV’: [’Harvested Volumes’, r"$Mm^3$"],
       ’vReg’: [’Regeneration Volumes’, r"$Mm^3$"],
       ’vol’: [’Forest Volumes’, r"$Mm^3$"],
       ’expReturns’: [’Expected returns’,’€/ha’,’forArea’, ’totalExpReturns’],
       ’forArea’: [’Forest area’,’ha’],
       ’harvestedArea’: [’Harvested area’,’ha’],
       ’regArea’: [’Regeneration area’,’ha’],
       ’STOCK_INV’: [’Carbon pool in inventoried forest resources’, r"$Mt CO_2$"],
       ’STOCK_EXTRA’: [’Carbon pool in non-inventoried forest resources (branches, roots)’, r"$Mt CO_2$"],
       ’STOCK_PRODUCTS’: [’Carbon pool in forest products’, r"$Mt CO_2$"],
       ’EM_ENSUB’: [’Cumulative emissions from energy substitution’, r"$Mt CO_2$"],
       ’EM_MATSUB’: [’Cumulative emissions from material substitution’, r"$Mt CO_2$"],
       ’EM_FOROP’: [’Cumulative emissions from forest operations’, r"$Mt CO_2$"],
  }

# =============================================================================
def printCharts():
  print "Printing charts.."

  title(’c’,’subsection’, "Carbon charts")
  plotCarbonChart([’constant’,’baseline’],’11000’,’’,’cbalance_overall’)
  plotCarbonChart([’baseline’,’Exp_0’,’Exp_1’],’11000’,’’,’cbalance_expectations’)
  plotCarbonChart([’baseline’,’Pr_C’,’Pr_U’],’11000’,’’,’cbalance_prices’)
  plotCarbonChart([’baseline’,’Ph_L’,’Ph_U’],’11000’,’’,’cbalance_ph_impact’)
```

```

# ==============================================================================
def printTables():
  print "Printing tables.."

  y2014_2060 =  [str(y) for y in range(2014,2061)] # [2014-2060]

  title(’t’,’section’, "Overall effect")
  printTable(’constant’,[’baseline’],[’expReturns’,’vReg’,’vol’,’hV’,’forArea’,’regArea’,’harvestedArea’],[’11000’],g.years,’\\texttt{Baseline} vs \\texttt{constant} [avg. 2007-2100]’,’cceffect_overall_vars_2007-2100_11000’)
  printTable(’constant’,[’baseline’],[’expReturns’,’vReg’,’vol’,’hV’,’forArea’,’regArea’,’harvestedArea’],[’11000’],[’2100’],’\\texttt{Baseline} vs \\texttt{constant} [2100]’,’cceffect_overall_vars_2100_11000’)
  printCarbonTable(’constant’,[’baseline’],’11000’, ’2007’, ’2100’, "\\ce{CO2} balance of \\texttt{baseline} scenario vs. \\texttt{constant} [yearly avg 2007-2100]",’cceffect_cbalance_2007-2100_11000’, True, True)
  printCarbonTable(’constant’,[’baseline’],’11000’, ’2013’, ’2020’, "\\ce{CO2} balance of \\texttt{baseline} scenario vs. \\texttt{constant} [yearly avg 2013-2020]",’cceffect_cbalance_2013-2020_11000’, True, True)

  title(’t’,’section’, "Sa on price, physical and expectation effects")
  printTable(’baseline’,[’Pr_C’,’Pr_U’,’Ph_L’,’Ph_U’,’Exp_0’,’Exp_1’],[’expReturns’,’vReg’,’vol’,’hV’,’forArea’,’regArea’,’harvestedArea’],[’11000’],g.years,’SA [avg. 2007-2100]’,’sa_vars_2007-2100_11000’, False)
  printCarbonTable(’baseline’,[’Pr_C’,’Pr_U’,’Ph_L’,’Ph_U’,’Exp_0’,’Exp_1’],’11000’, ’2007’, ’2100’, "Sensitivity analisys \\ce{CO2} balance [avg. 2007-2100]",’sa_cbalance_2007-2100_11000’, True, False)
  printTable(’baseline’,[’Pr_C’,’Pr_U’,’Ph_L’,’Ph_U’,’Exp_0’,’Exp_1’],[’expReturns’,’vReg’,’vol’,’hV’,’forArea’,’regArea’,’harvestedArea’],[’11000’],y2014_2060,’SA [avg. 2014-2060]’,’sa_vars_2014-2060_11000’,False)
  printCarbonTable(’baseline’,[’Pr_C’,’Pr_U’,’Ph_L’,’Ph_U’,’Exp_0’,’Exp_1’],’11000’, ’2014’, ’2060’, "Sensitivity analisys \\ce{CO2} balance [yearly avg. 2014-2060]",’sa_cbalance_2014-2060_11000’, True, False)
  printTable(’baseline’,[’Pr_C’,’Pr_U’,’Ph_L’,’Ph_U’,’Exp_0’,’Exp_1’],[’expReturns’,’vReg’,’vol’,’hV’,’forArea’,’regArea’,’harvestedArea’],[’11000’],[’2100’],’SA [2100]’,’sa_vars_2100_11000’,False)

  printCarbonTable(’baseline’,[’EOL_en_U’],’11000’, ’2007’, ’2100’, "\\ce{CO2} balance of \\texttt{EOL\\_en\\_U} scenario vs. \\texttt{baseline} [yearly avg 2007-2100]",’EOL_en_U_cbalance_2007-2100_11000’, True, True)


# ==============================================================================
def printAATables():
  print "Printing area allocation tables.."

# ==============================================================================
# EXECUTION ACTUALLY STARTS HERE.....
main()
```

5.19 /home/lobianco/git/ffsm_pp/data/output/output_parser_globals.py File Reference

Namespaces

- output_parser_globals

Variables

- list forIFiles = [ ]
- list prodIFiles = [ ]
- list carbonIFiles = [ ]
- dictionary scenarios = {}
- list years = [ ]
- bool printChartsFlag = False
- bool printTablesFlag = False
- bool printAATablesFlag = False
- string chartoutdir = 'charts'
- string tableoutdir = 'tables'
- string tablesmaster = '00_master_tables'
- string chartsmaster = '00_master_charts'
- string charttype = 'pdf'
- string sep = ';'
- dictionary countries
- dictionary regions

5.20 output_parser_globals.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

# 0 - parameters

# input data filenames
# input data expected format
# - scen;parName;country;region;forType;diamClass;year;value;

forIFiles    = []
prodIFiles   = []
carbonIFiles = []

scenarios = {}
years = []
printChartsFlag = False
printTablesFlag = False
printAATablesFlag = False

chartoutdir = 'charts'
tableoutdir = 'tables'
tablesmaster = '00_master_tables'
chartsmaster = '00_master_charts'
charttype = 'pdf'
sep = ';'


# OLD
#countries = {'11000': [['11042', '11061', '11072', '11025', '11026', '11052', '11024', '11021',
#  '11083', '11043', '11023', '11010', '11081', '11063', '11041', '11062',
#  '11030', '11051', '11022', '11053', '11082', '11071',],'France']}


#regions = {'11042': 'Alsace', '11061': 'Aquitaine', '11072': 'Auvergne', '11025': 'Basse-Normandie', '11026': 'Bourgogne',
#    '11052': 'Bretagne', '11024': 'Centre', '11021': 'Champagne-Ardenne', '11083': 'Corse', '11043': 'Franche-Comté',
#    '11023': 'Haute-Normandie', '11010': 'Île de France', '11081': 'Languedoc-Roussillon', '11063': 'Limousin',
#    '11041': 'Lorraine', '11062': 'Midi-Pyrénées', '11030': 'Nord - Pas-de-Calais', '11051': 'Pays de la Loire',
#    '11022': 'Picardie', '11053': 'Poitou-Charentes', '11082': 'Provence-Alpes-Côte d\'Azur', '11071': 'Rhône-Alpes',}

countries = {'FRA': [['AL (FR42)', 'AQ (FR61)', 'AU (FR72)', 'BN (FR25)', 'BO (FR26)', 'BR (FR52)', 'CE (FR24)', 'CA (FR21)',
             'CO (FR83)', 'FC (FR43)', 'HN (FR23)', 'IF (FR10)', 'LR (FR81)', 'LI (FR63)', 'LO (FR41)', 'MP (FR62)',
             'NP (FR30)', 'PL (FR51)', 'PI (FR22)', 'PC (FR53)', 'PA (FR82)', 'RA (FR71)'],'France']}

regions = {'AL (FR42)': 'Alsace', 'AQ (FR61)': 'Aquitaine', 'AU (FR72)': 'Auvergne', 'BN (FR25)': 'Basse-Normandie',
           'BO (FR26)': 'Bourgogne', 'BR (FR52)': 'Bretagne', 'CE (FR24)': 'Centre', 'CA (FR21)': 'Champagne-Ardenne',
           'CO (FR83)': 'Corse', 'FC (FR43)': 'Franche-Comté', 'HN (FR23)': 'Haute-Normandie', 'IF (FR10)': 'Île de France',
           'LR (FR81)': 'Languedoc-Roussillon', 'LI (FR63)': 'Limousin', 'LO (FR41)': 'Lorraine', 'MP (FR62)': 'Midi-Pyrénées',
           'NP (FR30)': 'Nord - Pas-de-Calais', 'PL (FR51)': 'Pays de la Loire', 'PI (FR22)': 'Picardie',
           'PC (FR53)': 'Poitou-Charentes', 'PA (FR82)': 'Provence-Alpes-Côte d\'Azur', 'RA (FR71)': 'Rhône-Alpes'}

# key: var short name
# value: turple with long name, unit and optionally variable to act for ponderation and name of aggregated variable. 20160815: added info if the ponderation variable is specific for the same ft (keywork: 'sameft') or global for all the forest types ('globalft')
# These should be called forVars
forVars = {'hV': ['Harvested volumes', r"$Mm^3$"],
       'vReg': ['Regeneration volumes', r"$Mm^3$"],
       'vol': ['Forest volumes', r"$Mm^3$"],
       'sumExpReturns': ['Sum of expected returns', r"€"],
       #'expReturns': ['Expected returns','€/ha','forArea', 'totalExpReturns','globalft'], # the script doesn't use sumExpReturns, it computes itself the value using the pounderation variable and put the value in this temporary variable . Then it does compute the total aggregate using this intermediate variable (as it should be).
       'expReturns': ['Expected returns','€/ha'],
       'forArea': ['Forest area','ha'],
       'harvestedArea': ['Harvested area','ha'],
       'regArea': ['Regeneration area','ha'],
       'STOCK_INV': ['Carbon pool in inventoried forest resources', r"$Mt CO_2$"],
       'STOCK_EXTRA': ['Carbon pool in non-inventoried forest resources (branches, roots)', r"$Mt CO_2$"],
       'STOCK_PRODUCTS': ['Carbon pool in forest products', r"$Mt CO_2$"],
       'EM_ENSUB': ['Cumulative emissions from energy substitution', r"$Mt CO_2$"],
       'EM_MATSUB': ['Cumulative emissions from material substitution', r"$Mt CO_2$"],
       ’EM_FOROP’: [’Cumulative emissions from forest operations’, r"$Mt CO_2$"],
           }
# key: var short name
# value: list with long name, unit, domain (either pp, tp or p) and optionally a variable to act for ponderation
prodVars = {’st’: [’Total supply’, r"$Mm^3$", ’p’],
            ’pl’: [’Local price’, r"$Mm^3$", ’p’, ’st’],
           }



spGroups = [’broadL_highF’,’broadL_mixedF’,’broadL_copp’,’con_highF’]
pProd = [’hardWRoundW’,’softWRoundW’,’pulpWFuelW’,’ashRoundW’]
tProd=[’fuelW’,’hardWSawnW’,’softWSawnW’,’plyW’,’pulpW’,’pannels’,’ashSawnW’,’ashPlyW’]




#key: human name
#value[0]: list of sp groups
#value[1]: chart line type
#value[2]: chart line width
#value[3]: (optional) alias in the data. If present, the input data will be converted to the name at input time
spAggregates = {’00_Total’: [[’broadL_highF’,’broadL_mixedF’,’broadL_copp’,’con_highF’],’-’,4,’’],
                ’01_Broadleaved’: [[’broadL_highF’,’broadL_mixedF’,’broadL_copp’],’--’,3,’broadL’],
                ’02_Coniferous’: [[’con_highF’],’:’,3,’con’]}

tvalue001  = [63.6567411629, 9.9248432009, 5.8409093097, 4.6040948714, 4.0321429836, 3.7074280213, 3.4994832974, 3.3553873313, 3.2498355416, 3.1692726726, 3.1058065155, 3.0545395894, 3.0122758387, 2.9768427344, 2.9467128835, 2.9207816224, 2.8982305197, 2.8784404727, 2.8609346065, 2.8453397098, 2.831359558, 2.8187560606, 2.8073356838, 2.7969395048, 2.7874358137, 2.7787145333, 2.7706829571, 2.7632624555, 2.7563859037, 2.7499956536, 2.7440419193, 2.738481482, 2.7332766424, 2.7283943671, 2.7238055892, 2.7194846305, 2.7154087215, 2.7115576019, 2.7079131835, 2.7044592674, 2.7011813036, 2.6980661862, 2.6951020792, 2.6922782657, 2.6895850194, 2.6870134922, 2.6845556179, 2.682204027, 2.6799519736, 2.6777932709]    # invt for alpha=0.01
tvalue0001 = [636.6192487687, 31.5990545764, 12.9239786367, 8.6103015814, 6.8688266259, 5.9588161788, 5.4078825209, 5.0413054334, 4.7809125859, 4.5868938587, 4.4369793382, 4.3177912836, 4.2208317277, 4.1404541127, 4.0727651959, 4.0149963272, 3.9651262721, 3.9216458251, 3.8834058526, 3.8495162749, 3.8192771643, 3.7921306717, 3.7676268043, 3.7453986193, 3.7251439497, 3.7066117435, 3.6895917135, 3.6739064007, 3.6594050195, 3.645958635, 3.6334563498, 3.6218022599, 3.6109130077, 3.6007157974, 3.5911467758, 3.5821497015, 3.5736748444, 3.5656780716, 3.5581200813, 3.5509657609, 3.544183643, 3.5377454453, 3.5316256778, 3.5258013065, 3.520251465, 3.5149572055, 3.5099012834, 3.5050679705, 3.5004428914, 3.4960128818] # invt for alpha=0.001




# -----------------------------------------------------------------------------
# global containers, don’t touch
idata = {}
odata = {}
x = []
sortedregions = []
products= pProd+tProd
```

5.21 /home/lobianco/git/ffsm_pp/data/output/output_parser_lib.py File Reference

Namespaces

- output_parser_lib

Functions

- def prepare_data ()
- def reset_output ()
- def plotMultivariable (scenarios_h, variables_h, region, title, filename, printLegend=True, fwidth=10, fheight=15)
- def plotCarbonChart (scenarios_h, region, title, filename)
- def plotLegend (scenarios_h, filename, title_h="")
- def plotVectorChart_inner (origin, end1, endt, xlabel, ylabel, filename, comp1_color='red', totcomp_color='blue', diffcomp_color='green')
- def printTable (ref_scenario, comparing_scenarios, variables_h, regions_h, years_h, title, filename, singleComparation=False, refYear=0)

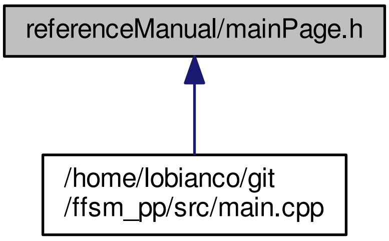
- def printAATable (ref_scenarios, comparing_scenarios, regions_h, years_h, title, filename, refYear=0)
- def printCarbonTable (ref_scenario, comparing_scenarios, region, year_start, year_end, title, filename, avg=False, singleComparation=True)
- def printTableRecord (cvar_label, d, el, nscen, valRScenario, valCScenarios, singleComparation)
- def title (cat, level, title)
- def text (cat, text_h)
- def myunicode (astring)

5.22 output_parser_lib.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import os, sys
import csv, math
from numba import jit
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import output_parser_globals as g

’’’ Scope of this script
- parse the pythia output to produce nice summarized tables
’’’


# ============================================================================
# jit decorator tells Numba to compile this function.
# The argument types will be inferred by Numba when function is called.
def prepare_data():
  #print ("Loading and preparing the data..")

  # A - creating empty dictionaries with just the keys..
  for country, data in g.countries.items():
    g.regions[country] = data[1] # add 11000: ’France’ to regions
  g.sortedregions = sorted(g.regions)
  #k = d.keys(); k.sort(). Use k = sorted(d)

  specieswithAggregates = g.spGroups
  specieswithAggregates.extend(g.spAggregates.keys())
  tempSpecieswithAggregates = specieswithAggregates
  #tempSpecieswithAggregates.append("") # attenction that python doesn not create a new variable, just alias the two
  tempSpGroups = g.spGroups
  tempSpGroups.append("")


  variablesWithAggregates = list(g.forVars.keys())
  for variable in g.forVars.keys():
    #’expReturns’: [’Expected returns’,’€/ha’,’forArea’,’totalExpReturns’,’globalft’],
    if len(g.forVars[variable]) >= 3:
      variablesWithAggregates.append(g.forVars[variable][3])

  for region in g.regions.keys():
    for variable in variablesWithAggregates:
      for scenario in g.scenarios.keys():
        for spGroup in tempSpecieswithAggregates:
          for year in g.years:
            key = region, variable, scenario, spGroup, year
            g.idata[key] = 0.0
  for region in g.regions.keys():
    for variable in variablesWithAggregates:
      for scenario in g.scenarios.keys():
        for spGroup in tempSpecieswithAggregates:
          key = region, variable, scenario, spGroup
          g.odata[key] = []
  for year in g.years:
    g.x.append(int(year))


  # B - loading data..
  for ifile in g.forIFiles:
    idata_raw = csv.DictReader(open(ifile, ’r’), delimiter=g.sep)
    for rec in idata_raw:
      # scen;parName;country;region;forType;diamClass;year;value;
      iForType  = rec[’forType’]
      if iForType == ’broadL’:
        debug = True
      for spAggregateKey, spAggregate in g.spAggregates.items():
        if (len(spAggregate) >= 3 and iForType ==  spAggregate[3]):
          iForType = spAggregateKey
          break
      key = rec[’region’],rec[’parName’],rec[’scen’],iForType,rec[’year’]
      if key in g.idata:
        g.idata[key] += float (rec[’value’])
  debug = g.idata
  for ifile in g.prodIFiles:
    idata_raw = csv.DictReader(open(ifile, ’r’), delimiter=g.sep)
    for rec in idata_raw:
      # scen;parName;country;region;prod;freeDim;year;value;
      key = rec[’region’],rec[’parName’],rec[’scen’],rec[’prod’],rec[’year’]
      if key in g.idata:
        g.idata[key] += float (rec[’value’])

  for ifile in g.carbonIFiles:
    #print (g.carbonIFiles)
    idata_raw = csv.DictReader(open(ifile, ’r’), delimiter=g.sep)
    for rec in idata_raw:
      # scen;parName;country;region;forType;diamClass;year;value;
      key = rec[’region’],rec[’balItem’],rec[’scen’],"",rec[’year’]
      #print key
      if key in g.idata:
        g.idata[key] += float (rec[’value’])
        #print (key)
        #print (g.idata[key])

  #exit(1)

  # C - creating aggregated data for variables that need to be pondered
#  for variable in g.forVars.keys():
#    #’expReturns’: [’Expected returns’,’€/ha’,’forArea’, ’totalExpReturns’],
#    if len(g.forVars[variable]) >= 3:
#      pondVariable = g.forVars[variable][2]
#      totalVariable = g.forVars[variable][3]
#      for region in g.regions.keys():
#        for scenario in g.scenarios.keys():
#          for spGroup in specieswithAggregates:
#            for year in g.years:
#              key = region, variable, scenario, spGroup, year
#              key_tvar = region, totalVariable, scenario, spGroup, year
#              if(g.forVars[variable][4] == ’sameft’):
#                  key_pvar = region, pondVariable, scenario, spGroup, year
#                  g.idata[key_tvar] = g.idata[key] * g.idata[key_pvar]
#              elif(g.forVars[variable][4] == ’globalft’):
#                  totalPvar = 0.0;
#                  for spGroup2 in g.spGroups:
#                      key_pvar = region, pondVariable, scenario, spGroup2, year
#                      totalPvar +=g.idata[key_pvar]
#                  g.idata[key_tvar] = g.idata[key] * totalPvar
#              else:
#                  print("Error, I don’t know how to handle this ponderation method: "+g.forVars[variable][4])
#                  exit(1)

  # D - performing various summing up..

  # summing up the specie aggregation
  for spAggregate, species in g.spAggregates.items():
    for region in g.regions.keys():
      for variable in variablesWithAggregates:
        if(variable != ’expReturns’ and variable != ’sumExpReturns’): # let’s skip these as the sumExpReturns at group/forest levels are already exogenously read as these are not the sums
         for scenario in g.scenarios.keys():
           for year in g.years:
             destKey = region, variable, scenario, spAggregate, year
             g.idata[destKey] = 0.0
             for specie in species[0]:
               varToBeSumKey = region, variable, scenario, specie, year
               g.idata[destKey] +=  g.idata[varToBeSumKey]

  # summing up to the country level..
  for country, regionsInTheCountry in g.countries.items():
    for variable in variablesWithAggregates:
      for scenario in g.scenarios.keys():
        for spGroup in tempSpGroups:
          for year in g.years:
            destKey = country, variable, scenario, spGroup, year
            g.idata[destKey] = 0.0
            for regionInTheCountry in regionsInTheCountry[0]:
              varToBeSumKey = regionInTheCountry, variable, scenario, spGroup, year
              g.idata[destKey] +=  g.idata[varToBeSumKey]

    # Correcting the country aggregation of expected returns
    for scenario in g.scenarios.keys():
      for spGroup in tempSpGroups:
        for year in g.years:
          countryForArea_key = country,’forArea’,scenario,’00_Total’,year
          countrySumExpReturns_key = country, 'sumExpReturns', scenario, spGroup, year
          target_key = country,'expReturns', scenario, spGroup, year
          g.idata[target_key] = g.idata[countrySumExpReturns_key]/ g.idata[countryForArea_key]

  # checking country aggregation, ok
  #for country, regionsInTheCountry in countries.iteritems():
    #print "country: " + country + " " + str(idata[country,'vol', 'vRegFixed', 'broadL_highF', '2006'])
    #for regionInTheCountry in regionsInTheCountry[0]:
      #print "region: " + regionInTheCountry + " " + str(idata[regionInTheCountry,'vol', 'vRegFixed', 'broadL_highF', '2006'])



  # testing specie aggregating
  #for spAggregate, species in spAggregates.iteritems():
    #print "aggregate: "+ spAggregate + " " + str(idata['11042','vol', 'vRegFixed', spAggregate, '2006'])
    #for specie in species[0]:
      #print "specieGroup: " + specie + " " + str(idata['11042','vol', 'vRegFixed', specie, '2006'])

#  # E - after all the summing up let's compute the poundered value
#  for variable in g.forVars.keys():
#    #'expReturns': ['Expected returns','€/ha','forArea', 'totalExpReturns'],
#    if len(g.forVars[variable]) >= 3:
#      pondVariable = g.forVars[variable][2]
#      totalVariable = g.forVars[variable][3]
#      for region in g.regions.keys():
#        for scenario in g.scenarios.keys():
#          for spGroup in specieswithAggregates:
#            for year in g.years:
#              key = region, variable, scenario, spGroup, year
#              key_pvar = region, pondVariable, scenario, spGroup, year
#              key_tvar = region, totalVariable, scenario, spGroup, year
#              g.idata[key] =  (g.idata[key_tvar] / g.idata[key_pvar]) if  g.idata[key_pvar] != 0 else 0

  # testing ponderation variables
  #for variable in variables.keys():
    #'expReturns': ['Expected returns','€/ha','forArea', 'totalExpReturns'],
    #if len(variables[variable]) >= 3:
      #pondVariable = variables[variable][2]
      #totalVariable = variables[variable][3]
      #print "Orig variable: " + variable + " " + str(idata['11000', variable, 'vRegFixed','Total', '2006'])
      #print "Pond variable: " + pondVariable + " " + str(idata['11000', pondVariable, 'vRegFixed', 'Total', '2006'])
      #print "Total variable: "  + totalVariable + " " + str(idata['11000', totalVariable, 'vRegFixed', 'Total', '2006'])

  # F - converting everything in years array
  for region in g.regions.keys():
    for variable in variablesWithAggregates:
      for scenario in g.scenarios.keys():
        for spGroup in tempSpecieswithAggregates:
          key = region, variable, scenario, spGroup
          for year in g.years:
            key_year = region, variable, scenario, spGroup, year
            g.odata[key].append(g.idata[key_year])

  # testing odata
  #print "idata[2005]: " + str(idata['11000', 'vol', 'vRegFixed','Total', '2005'])
  #print "idata[2006]: " + str(idata['11000', 'vol', 'vRegFixed','Total', '2006'])
  #print "odata: " + str(odata['11000', 'vol', 'vRegFixed','Total'])

# =============================================================================
def reset_output():
  # G - Reset latex files
  filename_t = g.tableoutdir+'/'+g.tablesmaster+'.tex'
  filename_c = g.chartoutdir+'/'+g.chartsmaster+'.tex'
  file_t = open(filename_t,'w')
  file_c = open(filename_c,'w')
  file_t.close()
  file_c.close()

# =============================================================================
def plotMultivariable(scenarios_h, variables_h, region, title, filename, printLegend=True, fwidth=10, fheight=15):

  nvar = len(variables_h)
  nscen = len(scenarios_h)
  #plt.figure(1)
  fig = plt.gcf()
  # suggested: fheight = (15/5)*nvar+0.2
  #if nvar == 1:
  #  fheight = 4
  #if nvar == 2:
  #  fheight = 8
  fig.set_size_inches(10,fheight) # 15 inches height is fine with 4 variables
  maintitle = myunicode(title)
  handles =[]
  labels = []
  #plt.suptitle(maintitle, fontsize=16, ha=’center’)
  for i in range(nvar):
    #plt.subplot(nvar,1,i+1)
    ax =fig.add_subplot(nvar,1,i+1)
    subplotTitle = myunicode(g.forVars[variables_h[i]][0])
    ylabel = myunicode(g.forVars[variables_h[i]][1])
    plt.title(subplotTitle)
    plt.ylabel(ylabel)
    for spGroup in sorted(g.spAggregates.keys()):
      for scenario in scenarios_h:
        serieName = myunicode(spGroup + " - " + scenario)
        serieColor = g.scenarios[scenario]
        serieLineType = g.spAggregates[spGroup][1]
        serieWidth = g.spAggregates[spGroup][2]
        #print serieName+ " - " + serieLineType + " - " + str(serieWidth)
        key = region, variables_h[i], scenario, spGroup
        y = g.odata[key]
        plt.plot(g.x, y, serieLineType, label=serieName, linewidth=serieWidth, color=serieColor)
    handles, labels = ax.get_legend_handles_labels()
  #plt.subplots_adjust(hspace=0.6)
  #handles, labels = ax.get_legend_handles_labels()
  #ax.legend(handles, labels, ncol=3, shadow=False, title="Legend")
  if printLegend:
    plt.figlegend(handles, labels, loc = ’lower center’, ncol=3, shadow=False, labelspacing=0., prop={’size’:12})
  #plt.savefig(chartoutdir+"/"+filename+"_"+region+"."+charttype, bbox_inches=’tight’, dpi=300)
  plt.savefig(g.chartoutdir+"/"+filename+"_"+region+"."+g.charttype, dpi=300)
  #plt.show()
  plt.close()

  omasterfilename = g.chartoutdir+’/’+g.chartsmaster+’.tex’
  omfile = open(omasterfilename,’a’)
  omfile.write("\\begin{figure}[htbp]\n")
  omfile.write("  \\centering\n")
  omfile.write("  \\caption{"+title+"}\n")
  omfile.write("  \\includegraphics[width=0.8\\textwidth]{\""+g.chartoutdir+"/"+filename+"_"+region+"\"}\n")
  omfile.write("  \\label{fig:"+filename+"}\n")
  omfile.write("\\end{figure}\n")
  omfile.close()

# =============================================================================
def plotCarbonChart(scenarios_h,region,title, filename):
#def plotMultivariable(scenarios_h, variables_h, region, title, filename, printLegend=True):


  cVariables = [
      [’Forest pool’, [’STOCK_INV’,’STOCK_EXTRA’],’:’,3,’#314004’],
      [’Wood products pool’, [’STOCK_PRODUCTS’],’--’,3,’#7f0021’],
      [’Net cumulative substitution effect’, [’EM_ENSUB’,’EM_MATSUB’,’EM_FOROP’],’-’,4,’#83caff’],
  ]

  nscen = len(scenarios_h)


  matplotlib.rcParams.update({’font.size’: 22})


  fig = plt.gcf()
  fig.set_size_inches(12,10)
  ylabel = myunicode("Gt CO2 eq")
  plt.title(myunicode(title))
  plt.ylabel(ylabel)

  totals = [[0]*len(g.x)]* nscen

  if nscen > 1: #normal line plots
    for idg, cGroup in enumerate(cVariables):
      for ids, scenario in enumerate(scenarios_h):
        grTotals =  [0]*len(g.x)
        #serieName = myunicode(cGroup[0] + " - " + scenario)
        serieName = "_"+myunicode(scenario) # not shown in legend
        if idg==2:
          serieName = myunicode(scenario)
        serieColor = g.scenarios[scenario]
        serieLineType = cGroup[2]
        serieWidth = cGroup[3]
        for var in cGroup[1]: # for idx, var in enumerate(cGroup[1]):
          key = region, var, scenario, ""
          varData = g.odata[key]
          grTotals = [x2+y for x2, y in zip(grTotals, varData)]

        totals[ids]  = [x3+y2 for x3, y2 in zip(totals[ids],grTotals)]
        y = [x4 / 1000 for x4 in totals[ids]]
        plt.plot(g.x, y, serieLineType, label=serieName, linewidth=serieWidth, color=serieColor)
  else: #area stacked plot
    fillColours = []
    y = []
    for cGroup in cVariables:
      y_local =  np.zeros(len(g.x))
      fillColour = cGroup[4]
      for var in cGroup[1]: # for idx, var in enumerate(cGroup[1]):
          key = region, var, scenarios_h[0], ""
          varData = np.array(g.odata[key])
          #y_local += varData # For some reasons this doesn’t work
          y_local = [t+(a/1000) for t, a in zip(y_local, varData)]
      y.append(y_local)
      fillColours.append(fillColour)
    for cGroup in reversed(cVariables):
      serieName = myunicode(cGroup[0])
      fillColour = cGroup[4]
      plt.plot([], [], color=fillColour, linewidth=4, label=serieName) # plotting emply data hack as stackplot doesn’t support the legend

    ax = fig.add_subplot(111)
    ax.stackplot(g.x, y, colors=fillColours, edgecolor = "none")
    ax.autoscale_view(’tight’)

  #plt.legend(loc=’lower right’, ncol=3, shadow=False, labelspacing=0., prop={’size’:12})
  plt.legend(loc=’lower right’, ncol=1, shadow=False, labelspacing=0., prop={’size’:14})
  #plt.ylim([0,18]) # This would scale the plot y axis to the desired ranges
  plt.savefig(g.chartoutdir+"/"+filename+"_"+region+"."+g.charttype, dpi=300)
  #plt.show()
  plt.close()

  omasterfilename = g.chartoutdir+’/’+g.chartsmaster+’.tex’
  omfile = open(omasterfilename,’a’)
  omfile.write("\\begin{figure}[htbp]\n")
  omfile.write("  \\centering\n")
  omfile.write("  \\caption{"+title+"}\n")
  omfile.write("  \\includegraphics[width=0.8\\textwidth]{\""+g.chartoutdir+"/"+filename+"_"+region+"\"}\n")
  omfile.write("  \\label{fig:"+filename+"}\n")
  omfile.write("\\end{figure}\n")
  omfile.close()

"""
      scenTotals
        y = odata[key]
  plt.plot(x, y, serieLineType, label=serieName, linewidth=serieWidth, color=serieColor)
    handles, labels = ax.get_legend_handles_labels()
  #plt.subplots_adjust(hspace=0.6)
  #handles, labels = ax.get_legend_handles_labels()
  #ax.legend(handles, labels, ncol=3, shadow=False, title="Legend")
  if printLegend:
    plt.figlegend(handles, labels, loc = ’lower center’, ncol=3, shadow=False, labelspacing=0., prop={’size’:12})
  #plt.savefig(chartoutdir+"/"+filename+"_"+region+"."+charttype, bbox_inches=’tight’, dpi=300)
  plt.savefig(chartoutdir+"/"+filename+"_"+region+"."+charttype, dpi=300)
  #plt.show()
  plt.close()

  omasterfilename = chartoutdir+’/’+chartsmaster+’.tex’
  omfile = open(omasterfilename,’a’)
  omfile.write("\\begin{figure}[htbp]\n")
  omfile.write("  \\centering\n")
  omfile.write("  \\caption{"+title+"}\n")
  omfile.write("  \\includegraphics[width=0.8\\textwidth]{"+chartoutdir+"/"+filename+"_"+region+"}\n")
  omfile.write("  \\label{fig:"+filename+"}\n")
  omfile.write("\\end{figure}\n")
  omfile.close()
  """

# ============================================================================
def plotLegend(scenarios_h, filename, title_h=""):
  nscen = len(scenarios_h)
  fig = plt.gcf()
  fheight = (15/15)*nscen+0.2
  fig.set_size_inches(10,fheight)
  #ax = plt.axes()
  #ax.set_axis_off()

  #fig = plt.figure()
  ax =fig.add_subplot(111)
  ax.set_axis_off()

  for spGroup in sorted(g.spAggregates.keys()):
    for scenario in scenarios_h:
      serieName = myunicode(spGroup + " - " + scenario)
      serieColor = g.scenarios[scenario]
      serieLineType = g.spAggregates[spGroup][1]
      serieWidth = g.spAggregates[spGroup][2]
      #print serieName+ " - " + serieLineType + " - " + str(serieWidth)
      dummyx = [1]
      dummyy = [1]
      plt.plot(dummyx, dummyy, serieLineType, label=serieName, linewidth=serieWidth, color=serieColor)
  handles, labels = ax.get_legend_handles_labels()
  ax.legend(handles, labels, ncol=3, shadow=False) # removed title=title_h
  plt.savefig(g.chartoutdir+"/"+filename+"."+g.charttype, bbox_inches=’tight’, pad_inches=0.1, dpi=300)
  #plt.show()
  plt.close()

  omasterfilename = g.chartoutdir+’/’+g.chartsmaster+’.tex’
  omfile = open(omasterfilename,’a’)
  omfile.write("\\begin{figure}[htbp]\n")
  omfile.write("  \\centering\n")
  omfile.write("  \\caption{"+title_h+"}\n")
  omfile.write("  \\includegraphics[width=0.8\\textwidth]{\""+g.chartoutdir+"/"+filename+"\"}\n")
  omfile.write("  \\label{fig:"+filename+"}\n")
  omfile.write("\\end{figure}\n")
  omfile.close()

#import matplotlib.pyplot as plt
#ax = plt.subplot()  #create the axes
#ax.set_axis_off()  #turn off the axis
#....  #do patches and labels
#ax.legend(patches, labels, ...)  #legend alone in the figure
#plt.show()

# ===============================================================================
def plotVectorChart_inner(origin,end1,endt,xlabel,ylabel,filename, comp1_color=’red’, totcomp_color=’blue’, diffcomp_color=’green’):
  ’’’
  Plot a 2-d vector difference
  # @params:
  # origin: x and y of the origin of the vectors
  # end1:  (x,y) coordinates of the ending of the first component vector
  # end2:  (x,y) coordinates of the ending of the total component of the vector
  # xlabel: xlabel
  # ylabel: ylabel
  # filename: filename
  # totcomp_color: color (English or #HTML_code) of the vector representing the total component
  # comp1_color: color (English or #HTML_code) of the vector representing the first component
  # diffcomp_color: color (English or #HTML_code) of the vector representing the difference component
  ’’’

  a   = plt.figure()
  ax  = plt.gca()
  fig = plt.gcf()
  flag_2d = True
  if(origin[0] == end1[0] == endt[0]):
    flag_2d = False;
    fig.set_size_inches(6,10)
  else:
    fig.set_size_inches(10,10)
  end2 = (endt[0]-end1[0]+origin[0],endt[1]-end1[1]+origin[1])
  minx = min(origin[0],end1[0],end2[0],endt[0])
  maxx = max(origin[0],end1[0],end2[0],endt[0])
  miny = min(origin[1],end1[1],end2[1],endt[1])
  maxy = max(origin[1],end1[1],end2[1],endt[1])
  centre = (((maxx-minx)/2)+minx,((maxy-miny)/2)+miny)

  # This allows to write a serie of arrows in one go, but didn’t got how in this case colours work
  #X  = (origin[0], origin[0], origin[0])
  #Y  = (origin[1], origin[1], origin[1])
  #X2 = (end1[0]-origin[0], endt[0]-origin[0], end2[0]-origin[0])
  #Y2 = (end1[1]-origin[1], endt[1]-origin[1], end2[1]-origin[1])
  #C  = (255,10,150) # ? colour codes, but didn’t got it
  # ax.quiver(X,Y,X2,Y2,Cangles=’xy’,scale_units=’xy’,scale=1, width=0.008)

  # Printing first component..
  ax.quiver(origin[0],origin[1],end1[0]-origin[0],end1[1]-origin[1],angles=’xy’,scale_units=’xy’,scale=1, width=0.008, color=comp1_color)
  # Printing total component..
  ax.quiver(origin[0],origin[1],endt[0]-origin[0],endt[1]-origin[1],angles=’xy’,scale_units=’xy’,scale=1, width=0.008, color=totcomp_color)
  # Printing diff component..
  ax.quiver(origin[0],origin[1],end2[0]-origin[0],end2[1]-origin[1],angles=’xy’,scale_units=’xy’,scale=1, width=0.008, color=diffcomp_color)

  x  = (end1[0],end2[0])
  y  = (end1[1],end2[1])
  x2 = (endt[0]-end1[0], endt[0]-end2[0])
  y2 = (endt[1]-end1[1], endt[1]-end2[1])

  if(flag_2d):
    ax.quiver(x,y,x2,y2,angles=’xy’,scale_units=’xy’,scale=1, width=0.005, color=’gray’)
    ax.set_xlim([minx- (centre[0]-minx)*0.4, maxx + (maxx-centre[0])*0.4])

  ax.set_ylim([miny- (centre[1]-miny)*0.4, maxy + (maxy-centre[1])*0.4])

  plt.xlabel(myunicode(xlabel))
  plt.ylabel(myunicode(ylabel))
  # Uncomment the following lines if you want to display instead of save the figure..
  #plt.draw()
  #plt.show()
  plt.savefig(filename, dpi=300, transparent=False, bbox_inches='tight', pad_inches=0.1)

# =============================================================================
def printTable(ref_scenario, comparing_scenarios, variables_h, regions_h, years_h, title, filename, singleComparation=False, refYear=0):
  """Print a LaTeX Table for variables variable_h comparing ref_scenario scenario vs coparing_scenarios.
  @param singleComparation: if True multiple comparing scenarios are treated as multiple replications of the same scenario and
  some basic stats are computed; if False they are all represented as diff from the ref_scenario.
  @param refYear: if 0 reference vs comparing scenarios are compared on the same year (or average of years if years_h has length > 1.).
  Otherwise the comparing scneario at year(s) years_h is compared with reference scenario at year refYear (useful to see the dynamic
  effects within a single scenario)
  """
  d = " & "
  el = " \\\\"
  label_comparing_scenario = "comparing scenarios"
  labels_comparing_scenarios = []
  nvar = len(variables_h)
  nscen = len(comparing_scenarios)
  nyears = len(years_h)
  nregions = len(regions_h)
  ncol = 4
  label_ref_scenario = ref_scenario.replace("_", "\\_")

  for comp_scenario in comparing_scenarios:
    labels_comparing_scenarios.append(comp_scenario.replace("_", "\\_"))

  if (singleComparation and nscen == 1):
    label_comparing_scenario = labels_comparing_scenarios[0]

  if (singleComparation):
    if nscen > 2:
      ncol = 5
  else:
    ncol = nscen+2 #+1 for the val label and +1 for the ref scenario

  oString = ""
  oString += "\\begin{table}[htbp]\n"
  oString += "\\begin{center}\n"
  oString += "\\begin{threeparttable}\n"
  oString += "\\centering\n"
  oString += "\\caption{"+title+"}\n"
  oString += "\\begin{footnotesize}\n"
  oString += "\\begin{tabularx}{\\textwidth}{l "
  for nc in range(1,ncol):
    oString += "r "
  oString += "}\n"
  oString += "\\hline\n"
  if (singleComparation):
    if nscen > 2:
      oString += d+label_ref_scenario+d+label_comparing_scenario+d+"difference"+d+"cv"+el+"\n"
    else:
      oString += d+label_ref_scenario+d+label_comparing_scenario+d+"difference"+el+"\n"
  else:
    oString += d+label_ref_scenario
    for label_comparing_scenarios in labels_comparing_scenarios:
      oString += d+label_comparing_scenarios
    oString += el+'\n'

  for region in regions_h:
    oString += "\\hline\n"
    if nregions > 1:
      oString += "\\multicolumn{"+str(ncol)+"}{l}{"+regions[region]+"}"+el+'\n'

    for variable in variables_h:
      oString += "\\multicolumn{"+str(ncol)+"}{l}{"+g.forVars[variable][0]+" (\\textit{"+g.forVars[variable][1]+"})}"+el+'\n'
      for spGroup in sorted(g.spAggregates.keys()):
        outSpGroup = spGroup.replace("_", "\\_")
        sumRScenario =   0
        sumCScenarios = [0] * nscen
        valRScenario = 0
        valCScenarios = [0] * nscen
        for year in years_h:
          rYear = str(refYear) if refYear else year # If we overrided the reference year we gonna pick it up here
          keyr = region, variable, ref_scenario, spGroup, rYear
          sumRScenario += g.idata[keyr]
          for s in range(nscen):
            keyc = region, variable, comparing_scenarios[s], spGroup, year
            sumCScenarios[s] += g.idata[keyc]
        valRScenario = sumRScenario/nyears
        for s in range(nscen):
          valCScenarios[s] = sumCScenarios[s]/nyears
        oString += printTableRecord("- "+outSpGroup, d, el, nscen, valRScenario,
valCScenarios, singleComparation)

  oString += "\\hline\n"
  oString += "\\end{tabularx}\n"
  oString += "\\end{footnotesize}\n"
  oString += "\\label{tab:"+filename+"}\n"
  if (singleComparation and nscen > 2):
    oString += "\\begin{tablenotes}\n"
    oString += "\\begin{footnotesize}\n"
    oString += "\\item [a] Significantly different from 0 at $\\alpha=0.01$\n"
    oString += "\\item [b] Significantly different from 0 at $\\alpha=0.001$\n"
    oString += "\\end{footnotesize}\n"
    oString += "\\end{tablenotes}\n"
  oString += "\\end{threeparttable}\n"
  oString += "\\end{center}\n"
  oString += "\\end{table}\n"

  ofilename = g.tableoutdir+’/’+filename+’.tex’
  ofile = open(ofilename,’w’)
  ofile.write(oString)
  ofile.close()

  omasterfilename = g.tableoutdir+’/’+g.tablesmaster+’.tex’
  omfile = open(omasterfilename,’a’)
  omfile.write("\\input{\""+g.tableoutdir+’/’+filename+".tex\"}\n")
  omfile.close()

# =============================================================================
def printAATable(ref_scenarios, comparing_scenarios, regions_h, years_h, title, filename,
refYear=0) :
#def printTable(ref_scenario, comparing_scenarios, variables_h, regions_h, years_h, title, filename):
#printAATable([’cc1’,’cc1_nospvar’,’cc2’,’cc2_nospvar’,’cc3’,’cc3_nospvar’,’cc3’,’cc3_nospvar’],[’bau’,’bau_nospvar’,’ba
 allocation [% variation over bau]’, ’area_allocation’)
  d = " & "
  el = " \\\\"

  scenario_labels = []
  nscen = len(ref_scenarios)
  nscen_comp = len(comparing_scenarios)
  if nscen != nscen_comp:
    print ("Error in printAATable: number of comparing vs reference scenarios must be the same !")
    exit(1)
  nyears = len(years_h)
  nregions = len(regions_h)
  ntotcol = nscen+1
  for scenario in comparing_scenarios:
    scenario_labels.append(scenario.replace("_", "\\_"))


  oString = ""
  oString += "\\begin{table}[htbp]\n"
  oString += "\\begin{center}\n"
  oString += "\\begin{threeparttable}\n"
  oString += "\\centering\n"
  oString += "\\caption{"+title.replace("_", "\\_").replace("%", "\\%")+"}\n"
  oString += "\\begin{footnotesize}\n"
  oString += "\\begin{tabularx}{\\textwidth}{l "
  for i in range(nscen):
    oString += " r"
  oString += "}\n"

  oString += "\\hline\n"
  oString += "Region"
  for scenario in scenario_labels:
    oString += d+scenario
  oString += el+’\n’
  for spGroup in sorted(g.spAggregates.keys()):
    oString += "\\multicolumn{"+str(ntotcol)+"}{l}{"+spGroup.replace("_", "\\_")+"}"+el+’\n’
    for region in regions_h:
      oString += g.regions[region]
      for s in range(len(comparing_scenarios)):
        sum_value_b = 0.0
        sum_value_c = 0.0
        for year in years_h:
          rYear = str(refYear) if refYear else year # If we overrided the reference year we gonna pick it
 up here
          key_b = region, ’forArea’, ref_scenarios[s], spGroup, rYear
          key_c = region, ’forArea’, comparing_scenarios[s], spGroup, year
          sum_value_b += g.idata[key_b]
          sum_value_c += g.idata[key_c]
        reldiff = (100*(sum_value_c-sum_value_b)/sum_value_b) if  sum_value_b != 0 else 0
        oString +=  d+"%+0.3f"%(reldiff)
      oString += el+’\n’


  oString += "\\hline\n"
  oString += "\\end{tabularx}\n"
  oString += "\\end{footnotesize}\n"
  oString += "\\label{tab:"+filename+"}\n"
  oString += "\\end{threeparttable}\n"
  oString += "\\end{center}\n"
  oString += "\\end{table}\n"

  ofilename = g.tableoutdir+’/’+filename+’.tex’
  ofile = open(ofilename,’w’)
  ofile.write(oString)
  ofile.close()

  omasterfilename = g.tableoutdir+’/’+g.tablesmaster+’.tex’
  omfile = open(omasterfilename,’a’)
  omfile.write("\\input{\""+g.tableoutdir+’/’+filename+".tex\"}\n")
  omfile.close()

# =============================================================================
def printCarbonTable(ref_scenario, comparing_scenarios, region, year_start, year_end, title, filename, avg=False, singleComparation=True ) :
  #Print carbon balance
  # @params:
  # avg:               true  => output is the yearly average in the period,
  #                    false => output is the difference between year_start and year_end
  # singleComparation: true  => comparing scenarios are seens as repetition of a unique scenario, hence stats on their variance is performed,
  #                    false => each comparing scenarios is presented indipendently
  d = " & "
  el = " \\\\"

  cvariables = [
      [’Pools’, "- Total pools", [
         [’STOCK_INV’,"- Inventoried forest pool"],
         [’STOCK_EXTRA’,"- Extra forest pool (branches and roots)"],
         [’STOCK_PRODUCTS’,"- Wood products pool"]
         ]],
      [’Emissions’, "- Net substitution",
        [[’EM_ENSUB’,"- Energy substitution"],
         [’EM_MATSUB’,"- Material substitution"],
         [’EM_FOROP’,"- Emissions from forest operations"]
         ]],
  ]

  label_comparing_scenario = "comparing scenarios"
  labels_comparing_scenarios = []
  nscen = len(comparing_scenarios)
  nyears = (int(year_end) - int(year_start) + 1) if avg else 1
  ncol = 4
  label_ref_scenario = ref_scenario.replace("_", "\\_")

  for comp_scenario in comparing_scenarios:
    labels_comparing_scenarios.append(comp_scenario.replace("_", "\\_"))

  if (singleComparation and nscen == 1):
    label_comparing_scenario = labels_comparing_scenarios[0]

  if (singleComparation):
    if nscen > 2:
      ncol = 5
  else:
    ncol = nscen+2

  oString = ""
  oString += "\\begin{table*}[!htbp]\n"
  oString += "\\begin{center}\n"
  oString += "\\begin{threeparttable}\n"
  oString += "\\centering\n"
  oString += "\\caption{"+title+"}\n"
  oString += "\\begin{footnotesize}\n"
  oString += "\\begin{tabularx}{\\textwidth}{l "
  for nc in range(1,ncol):
    oString += "r "
  oString += "}\n"
  oString += "\\hline\n"

  if (singleComparation):
    if nscen > 2:
      oString += d+"\\texttt{"+label_ref_scenario+"}"+d+"\\texttt{"+label_comparing_scenario+"}"+d+"difference"+d+"cv"+el+"\n"
    else:
      oString += d+"\\texttt{"+label_ref_scenario+"}"+d+"\\texttt{"+label_comparing_scenario+"}"+d+" difference"+el+"\n"
  else:
    oString += d+label_ref_scenario
    for label_comparing_scenarios in labels_comparing_scenarios:
      oString += d+label_comparing_scenarios
    oString += el+'\n'

  if(nyears > 1):
    oString += "\\multicolumn{"+str(ncol)+"}{l}{Carbon balance ($Mt~ \ce{CO2}eq.~y^{-1}$)}"+el+'\n'
  else:
    oString += "\\multicolumn{"+str(ncol)+"}{l}{Carbon balance ($Mt~ \ce{CO2}eq.)$}"+el+'\n'

  # Total totals..
  totSumValRScenario = 0
  totSumValCScenarios = [0] * nscen
  for vargroup in cvariables:
    # Group totals..
    grSumValRScenario = 0
    grSumValCScenarios = [0] * nscen
    oString += "\\multicolumn{"+str(ncol)+"}{l}{"+vargroup[0]+"}"+el+'\n'
    # Working on the single variables..
    for cvar in vargroup[2]:
      cvar_name           = cvar[0]
      cvar_label          = cvar[1]
      valRScenario        = (g.idata[region, cvar_name, ref_scenario, "", year_end]-g.idata[region, cvar_name, ref_scenario, "", year_start])/nyears
      grSumValRScenario  += valRScenario
      totSumValRScenario += valRScenario
      valCScenarios       = [0] * nscen

      for s in range(nscen):
        valCScenarios[s] = (g.idata[region, cvar_name, comparing_scenarios[s], "", year_end]-g.idata[region, cvar_name, comparing_scenarios[s], "", year_start])/nyears
        grSumValCScenarios[s] += valCScenarios[s]
        totSumValCScenarios[s]  += valCScenarios[s]
      oString += printTableRecord(cvar_label, d, el, nscen, valRScenario, valCScenarios, singleComparation)
    oString += printTableRecord(vargroup[1], d, el, nscen, grSumValRScenario, grSumValCScenarios,singleComparation)
  oString += printTableRecord("Total \ce{CO2} balance", d, el, nscen, totSumValRScenario, totSumValCScenarios,singleComparation)

  oString += "\\hline\n"
  oString += "\\end{tabularx}\n"
  oString += "\\end{footnotesize}\n"
  oString += "\\label{tab:"+filename+"}\n"
  if (singleComparation and nscen > 2):
    oString += "\\begin{tablenotes}\n"
    oString += "\\begin{footnotesize}\n"
    oString += "\\item [a] Significantly different from 0 at $\\alpha=0.01$\n"
    oString += "\\item [b] Significantly different from 0 at $\\alpha=0.001$\n"
    oString += "\\end{footnotesize}\n"
    oString += "\\end{tablenotes}\n"
  oString += "\\end{threeparttable}\n"
  oString += "\\end{center}\n"
  oString += "\\end{table*}\n"

  ofilename = g.tableoutdir+'/'+filename+'.tex'
  ofile = open(ofilename,'w')
  ofile.write(oString)
  ofile.close()

  omasterfilename = g.tableoutdir+'/'+g.tablesmaster+'.tex'
  omfile = open(omasterfilename,'a')
  omfile.write("\\input{\""+g.tableoutdir+'/'+filename+".tex\"}\n")
  omfile.close()
# =============================================================================
def printTableRecord(cvar_label, d, el, nscen, valRScenario, valCScenarios, singleComparation):

  oString = ""
  if singleComparation:
    avgCScenarios = sum(valCScenarios) / float(nscen)
    scenarioDiff = avgCScenarios-valRScenario
    scenarioRelativeDiff = 100 * scenarioDiff/valRScenario if valRScenario else 0.0
    if nscen > 2:
      significance = ""
      qdiffCScenarios = [0] * nscen
      sumqdiffCScenarios = 0
      for s in range(nscen):
        qdiffCScenarios[s] = (valCScenarios[s] - avgCScenarios)**2.0
        sumqdiffCScenarios += qdiffCScenarios[s]
      sd = (sumqdiffCScenarios/(nscen-1))**0.5
      t = abs(scenarioDiff)*nscen**0.5/sd if sd>0.0 else 0.0
      cv = 100.0 * sd/abs(avgCScenarios) if abs(avgCScenarios)> 0.0 else 0.0
      if t >= g.tvalue001[nscen-1-1]:
        significance = '$^a$'
      if t >= g.tvalue0001[nscen-1-1]:
        significance = '$^b$'
      oString += cvar_label+d+"%0.3f"%(valRScenario)+d+"%0.3f"%(avgCScenarios)+d+"%0.3f"%(scenarioDiff)+significance+' ('+"%0.3f"%(scenarioRelativeDiff)+'\\%)'+d+"%0.2f"%(cv)+' \\%'+el+'\n'
    else:
      oString += cvar_label+d+"%0.3f"%(valRScenario)+d+"%0.3f"%(avgCScenarios)+d+"%0.3f"%(scenarioDiff)+' ('+"%0.2f"%(scenarioRelativeDiff)+'\\%)'+el+'\n'
  else:
    oString += cvar_label+d+"%0.3f"%(valRScenario)
    for valCScenario in valCScenarios:
      scenarioDiff = valCScenario-valRScenario
      scenarioRelativeDiff = 100 * scenarioDiff/valRScenario if valRScenario else 0.0
      oString += d+"%0.2f"%(scenarioRelativeDiff)+'\\%'
    oString += el + '\n'
  return oString



# ============================================================================
def title (cat, level, title):
  filename = ""
  if cat == 't':
    filename = g.tableoutdir+'/'+g.tablesmaster+'.tex'
  elif cat == 'c':
    filename = g.chartoutdir+'/'+g.chartsmaster+'.tex'
  else:
    print ("Error in printTable: not know where to print the title !")
    exit(1)
  file = open(filename,'a')

  file.write("\n\\clearpage\n")
  file.write("\\"+level+"{"+title+"}\n")
  file.close()

# ============================================================================
def text(cat, text_h):
  filename = ""
  if cat == 't':
    filename = g.tableoutdir+'/'+g.tablesmaster+'.tex'
  elif cat == 'c':
    filename = g.chartoutdir+'/'+g.chartsmaster+'.tex'
  else:
    print ("Error in text: not know where to print the title !")
    exit(1)
  file = open(filename,'a')
  file.write(text_h+"\n")
  file.close()

# ============================================================================
def myunicode(astring):
  if sys.version_info < (3, 0):
    return unicode(astring, 'utf_8')
  else:
    return astring

```

5.23 referenceManual/mainPage.h File Reference

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



5.24 mainPage.h

```
/*!

   \mainpage FFSM++ Reference Manual (doxygen-generated)
  <p>&nbsp;<p>&nbsp;
  <p>This is the Reference Manual of <a href="http://www.ffsm-project.org">FFSM++</a>.
  <br>It contains detailed developer information on the C++ version of the model retrieved automatically
 from the latest version of the
  source code (updated daily).
  <br>It includes class description, class members, collaboration and caller graphs, as well as the full
 source code.
  <br>Developers can browse the GIT code from its <a href="https://github.com/LEFNancy/ffsm_pp">github web
 interface</a>.
  <br>Access to git is restricted as it included some input data for which we do not hold copyright and we
 can't hence redistribute.
  <br>If you need access to the source code in a more convenient form (e.g. a zip archive) or to a
 "cleaned-up" version of the input file
  please just drop <a href="http://ffsm-project.org/wiki/en/team/home#current_team">us</a> an email.

*/
```

5.25 /home/lobianco/git/ffsm_pp/ffsm.pro File Reference

5.26 /home/lobianco/git/ffsm_pp/ffsm.pro

```
SUBDIRS += src
TEMPLATE = subdirs




```

5.27 /home/lobianco/git/ffsm_pp/NEWS File Reference

5.28 /home/lobianco/git/ffsm_pp/NEWS

```
FFSM++ - French Forest Sector model
Info: http://ffsm-project.org

GIT commit logs: http://ffsm-project.org/wiki/en/dev/gitlogfull

***** NEWS *****

20150203 FFSM++ goes open-source !









```

5.29 /home/lobianco/git/ffsm_pp/README File Reference

5.30 /home/lobianco/git/ffsm_pp/README

```
*** To compile and install: ***

1) qmake  (or qmake-qt5)
2) make
3) ./ffsm

Notes:
```

```
- a project file for QtCreator is attahced. Hovewer you can use whatsoever IDE to work with the project. The interface file was generated using QTDesigner;
- you need the Qt5 development library to compile this program as well as the GLPK library.
- on some Linux distros you have to use qmake-qt5 in order to use Qt5 instead of the "old" Qt3/Qt4 libraries.
- detailed compiling instructions for both Linux and Windows are available on the http://ffsm-project.org web site

*** To use: ***
Please refer to the User Manual (http://ffsm-project.org/wiki/en/dev/installation).


***********************************************
Documentation (reference manual, user manual,
contributed wiki doc, community support) is at:
http://ffsm-project.org
***********************************************


```

5.31 /home/lobianco/git/ffsm_pp/run_single_scenario.sh File Reference

5.32 /home/lobianco/git/ffsm_pp/run_single_scenario.sh

```
#!/bin/bash

#---------------------------------------
#  Shell script to run a single ffsm scenario, where the scenario name is the first argument and input file is the second (optional) argument.
# e.g. ./run_single_scenario.sh 'data/ffsmInput_2015_wdulef.ods' 'baseline'
#---------------------------------------

if [ $# -eq 2 ]
  then
    ./ffsm -c -s $2 -i $1 > logs/${2}.txt
    echo "Ended running scenario" $2 "on input file" $1
  else
   if [ $# -eq 1 ]
    then
     ./ffsm -c -s $1 > logs/${1}.txt
     echo "Ended running scenario" $1
    else
     echo "ERROR: this script must be called with either 1 argument (scenario name) or 2 arguments (input file, scenario name)"
   fi
fi




```

5.33 /home/lobianco/git/ffsm_pp/runscenarios.sh File Reference

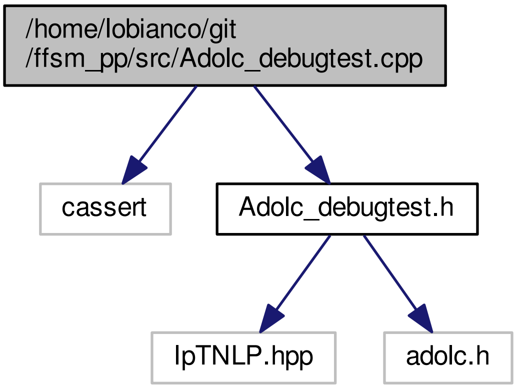
5.34 /home/lobianco/git/ffsm_pp/runscenarios.sh

```
#!/bin/bash

#---------------------------------------
#  Shell script to run ffsm scenarios
#---------------------------------------

# Safe parallel..
./ffsm -c -s scenarioName1 > logs/scenarioName1.txt &
./ffsm -c -s scenarioName2 > logs/scenarioName2.txt &


# Running the same scenario (e.g. for repetitions) in parallel is safe as long as newRandomSeed
# is set to true and outputSingleFile is set to false..
for i in {1..30}
do
   ./ffsm -c -s randomSpace 1> /dev/null 2> /dev/null &
done

# A better approach to run scenarios in parallel is using GNU parallel: you can set the maximum
# number of processes and then the jobs are put in a queue.
# In that case run this script as:
# parallel --jobs <n of jobs> -a runscenarios.sh
# and put something like this in the script

./run_single_scenario.sh 'scenarioName1'
./run_single_scenario.sh 'scenarioName2'
./run_single_scenario.sh 'inputFile1' 'scenarioName3'
./run_single_scenario.sh 'inputFile2' 'scenarioName4'



```

5.35 /home/lobianco/git/ffsm_pp/src/Adolc_debugtest.cpp File Reference

`#include <cassert>`
`#include "Adolc_debugtest.h"`

Include dependency graph for Adolc_debugtest.cpp:



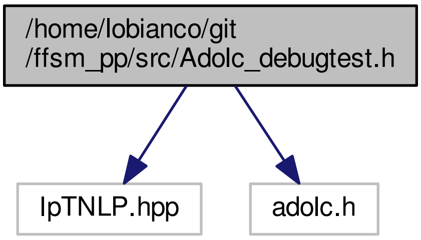
5.36 Adolc_debugtest.cpp

```
/*----------------------------------------------------------------------------
 ADOL-C -- Automatic Differentiation by Overloading in C++
 File:     ADOL-C_NLP.cpp
 Revision: $$
 Contents: class myADOLC_NPL for interfacing with Ipopt

 Copyright (c) Andrea Walther

 This file is part of ADOL-C. This software is provided as open source.
 Any use, reproduction, or distribution of the software constitutes
 recipient's acceptance of the terms of the accompanying license file.

 This code is based on the file  MyNLP.cpp contained in the Ipopt package
 with the authors:  Carl Laird, Andreas Waechter
----------------------------------------------------------------------------*/

/** C++ Example NLP for interfacing a problem with IPOPT and ADOL-C.
 *  MyADOL-C_NLP implements a C++ example showing how to interface
 *  with IPOPT and ADOL-C through the TNLP interface. This class
 *  implements the Example 5.1 from "Sparse and Parially Separable
 *  Test Problems for Unconstrained and Equality Constrained
 *  Optimization" by L. Luksan and J. Vlcek ignoring sparsity.
 *
 *  no exploitation of sparsity !!
 *
 */
#include <cassert>

#include "Adolc_debugtest.h"

using namespace Ipopt;

/* Constructor. */
MyADOLC_NLP::MyADOLC_NLP()
{}

MyADOLC_NLP::~MyADOLC_NLP(){}

bool MyADOLC_NLP::get_nlp_info(Index& n, Index& m, Index& nnz_jac_g,
                         Index& nnz_h_lag, IndexStyleEnum& index_style)
{
  n = 20;

  m = n-2;

  // in this example the jacobian is dense. Hence, it contains n*m nonzeros
  nnz_jac_g = n*m;

  // the hessian is also dense and has n*n total nonzeros, but we
  // only need the lower left corner (since it is symmetric)
  nnz_h_lag = n*(n-1)/2+n;

  generate_tapes(n, m);

  // use the C style indexing (0-based)
  index_style = C_STYLE;

  return true;
}

bool MyADOLC_NLP::get_bounds_info(Index n, Number* x_l, Number* x_u,
                            Index m, Number* g_l, Number* g_u)
{
  // none of the variables have bounds
  for (Index i=0; i<n; i++) {
    x_l[i] = -1e20;
    x_u[i] =  1e20;
  }

  // Set the bounds for the constraints
  for (Index i=0; i<m; i++) {
    g_l[i] = 0;
    g_u[i] = 0;
  }

  return true;
}

bool MyADOLC_NLP::get_starting_point(Index n, bool init_x, Number* x,
                               bool init_z, Number* z_L, Number* z_U,
                               Index m, bool init_lambda,
                               Number* lambda)
{
  // Here, we assume we only have starting values for x, if you code
  // your own NLP, you can provide starting values for the others if
  // you wish.
  assert(init_x == true);
  assert(init_z == false);
  assert(init_lambda == false);

  // set the starting point
  for (Index i=0; i<n/2; i++) {
    x[2*i] = -1.2;
    x[2*i+1] = 1.;
  }
  if (n != 2*(n/2)) {
    x[n-1] = -1.2;
  }

  return true;
}

template<class T> bool  MyADOLC_NLP::eval_obj(Index n, const T *x, T& obj_value)
{
  T a1, a2;
  obj_value = 0.;
  for (Index i=0; i<n-1; i++) {
    a1 = x[i]*x[i]-x[i+1];
    a2 = x[i] - 1.;
    obj_value += 100.*a1*a1 + a2*a2;
  }

  return true;
}

template<class T> bool  MyADOLC_NLP::eval_constraints(Index n, const T *x,
Index m, T* g)
{
  for (Index i=0; i<m; i++) {
    g[i] = 3.*pow(x[i+1],3.) + 2.*x[i+2] - 5.
           + sin(x[i+1]-x[i+2])*sin(x[i+1]+x[i+2]) + 4.*x[i+1]
           - x[i]*exp(x[i]-x[i+1]) - 3.;
  }

  return true;
}

//***************************************************************************
//
//
//          Nothing has to be changed below this point !!
//
//
//***************************************************************************


bool MyADOLC_NLP::eval_f(Index n, const Number* x, bool new_x, Number& obj_value)
{
  eval_obj(n,x,obj_value);

  return true;
}

bool MyADOLC_NLP::eval_grad_f(Index n, const Number* x, bool new_x, Number* grad_f)
{

  gradient(tag_f,n,x,grad_f);

  return true;
}

bool MyADOLC_NLP::eval_g(Index n, const Number* x, bool new_x, Index m, Number* g)
{

  eval_constraints(n,x,m,g);

  return true;
}

bool MyADOLC_NLP::eval_jac_g(Index n, const Number* x, bool new_x,
                       Index m, Index nele_jac, Index* iRow, Index *jCol,
                       Number* values)
{
  if (values == NULL) {
    // return the structure of the jacobian,
    // assuming that the Jacobian is dense

    Index idx = 0;
    for(Index i=0; i<m; i++)
      for(Index j=0; j<n; j++)
    {
      iRow[idx] = i;
      jCol[idx++] = j;
    }
 }
  else {
    // return the values of the jacobian of the constraints

    jacobian(tag_g,m,n,x,Jac);

    Index idx = 0;
    for(Index i=0; i<m; i++)
      for(Index j=0; j<n; j++)
      values[idx++] = Jac[i][j];

  }

  return true;
}

bool MyADOLC_NLP::eval_h(Index n, const Number* x, bool new_x,
                   Number obj_factor, Index m, const Number* lambda,
                   bool new_lambda, Index nele_hess, Index* iRow,
                   Index* jCol, Number* values)
{
  if (values == NULL) {
    // return the structure. This is a symmetric matrix, fill the lower left
    // triangle only.

    // the hessian for this problem is actually dense
    Index idx=0;
    for (Index row = 0; row < n; row++) {
      for (Index col = 0; col <= row; col++) {
        iRow[idx] = row;
        jCol[idx] = col;
        idx++;
      }
    }

    assert(idx == nele_hess);
  }
  else {
    // return the values. This is a symmetric matrix, fill the lower left
    // triangle only

    for(Index i = 0; i<n ; i++)
      x_lam[i] = x[i];
    for(Index i = 0; i<m ; i++)
      x_lam[n+i] = lambda[i];
    x_lam[n+m] = obj_factor;

    hessian(tag_L,n+m+1,x_lam,Hess);

    Index idx = 0;

    for(Index i = 0; i<n ; i++)
      {
    for(Index j = 0; j<=i ; j++)
      {
        values[idx++] = Hess[i][j];
      }
      }
  }

  return true;
}

void MyADOLC_NLP::finalize_solution(SolverReturn status,
                              Index n, const Number* x, const Number* z_L, const Number* z_U,
                              Index m, const Number* g, const Number* lambda,
                              Number obj_value,
                  const IpoptData* ip_data,
                  IpoptCalculatedQuantities* ip_cq)
{

  printf("\n\nObjective value\n");
  printf("f(x*) = %e\n", obj_value);

// Memory deallocation for ADOL-C variables

  delete[] x_lam;

  for(Index i=0;i<m;i++)
    delete[] Jac[i];
  delete[] Jac;

  for(Index i=0;i<n+m+1;i++)
    delete[] Hess[i];
  delete[] Hess;
}


//***************    ADOL-C part ***********************************

void MyADOLC_NLP::generate_tapes(Index n, Index m)
{
  Number *xp    = new double[n];
  Number *lamp  = new double[m];
  Number *zl    = new double[m];
  Number *zu    = new double[m];

  adouble *xa   = new adouble[n];
  adouble *g    = new adouble[m];
  adouble *lam  = new adouble[m];
  adouble sig;
  adouble obj_value;

  double dummy;

  Jac = new double*[m];
  for(Index i=0;i<m;i++)
    Jac[i] = new double[n];

  x_lam   = new double[n+m+1];

  Hess = new double*[n+m+1];
  for(Index i=0;i<n+m+1;i++)
    Hess[i] = new double[i+1];

  get_starting_point(n, 1, xp, 0, zl, zu, m, 0, lamp);

  trace_on(tag_f);

    for(Index i=0;i<n;i++)
      xa[i] <<= xp[i];

    eval_obj(n,xa,obj_value);

    obj_value >>= dummy;

  trace_off();

  trace_on(tag_g);

    for(Index i=0;i<n;i++)
      xa[i] <<= xp[i];

    eval_constraints(n,xa,m,g);


    for(Index i=0;i<m;i++)
      g[i] >>= dummy;

  trace_off();

   trace_on(tag_L);

    for(Index i=0;i<n;i++)
      xa[i] <<= xp[i];
    for(Index i=0;i<m;i++)
      lam[i] <<= 1.0;
    sig <<= 1.0;

    eval_obj(n,xa,obj_value);

    obj_value *= sig;
    eval_constraints(n,xa,m,g);

    for(Index i=0;i<m;i++)
      obj_value += g[i]*lam[i];

    obj_value >>= dummy;

  trace_off();

  delete[] xa;
  delete[] xp;
  delete[] g;
  delete[] lam;
  delete[] lamp;
  delete[] zu;
  delete[] zl;

}
```

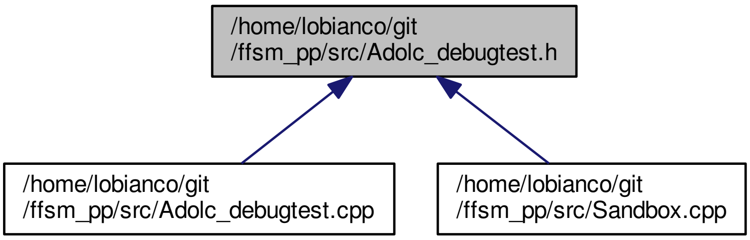
5.37 /home/lobianco/git/ffsm_pp/src/Adolc_debugtest.h File Reference

```
#include "IpTNLP.hpp"
#include <adolc.h>
```

Include dependency graph for Adolc_debugtest.h:



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



Classes

- class MyADOLC_NLP

Macros

- #define tag_f 1
- #define tag_g 2
- #define tag_L 3

5.37.1 Macro Definition Documentation

5.37.1.1 #define tag_f 1

Definition at line 31 of file Adolc_debugtest.h.

Referenced by MyADOLC_NLP::eval_grad_f(), Opt::eval_grad_f(), MyADOLC_NLP::generate_tapes(), and Opt↩ ::generate_tapes().

5.37.1.2 #define tag_g 2

Definition at line 32 of file Adolc_debugtest.h.

Referenced by Opt::calculateSparsityPatternJ(), MyADOLC_NLP::eval_jac_g(), Opt::eval_jac_g(), MyADOLC_NLP::generate_tapes(), and Opt::generate_tapes().

5.37.1.3 #define tag_L 3

Definition at line 33 of file Adolc_debugtest.h.

Referenced by Opt::calculateSparsityPatternH(), MyADOLC_NLP::eval_h(), Opt::eval_h(), MyADOLC_NLP::generate_tapes(), and Opt::generate_tapes().

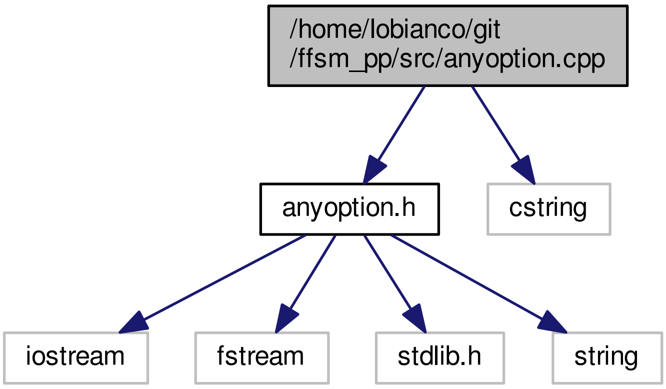
5.38 Adolc_debugtest.h

```
00001 /*----------------------------------------------------------------------------
00002  ADOL-C -- Automatic Differentiation by Overloading in C++
00003  File:     ADOL-C_NLP.hpp
00004  Revision: $$
00005  Contents:  class myADOL-C_NPL for interfacing with Ipopt
00006 
00007  Copyright (c) Andrea Walther
00008 
00009  This file is part of ADOL-C. This software is provided as open source.
00010  Any use, reproduction, or distribution of the software constitutes
00011  recipient's acceptance of the terms of the accompanying license file.
00012 
00013  This code is based on the file  MyNLP.hpp contained in the Ipopt package
00014  with the authors:  Carl Laird, Andreas Waechter
00015 ----------------------------------------------------------------------------*/
00016 
00017 //*************************************************************************
00018 //
00019 //
00020 //         Nothing has to be changed in this file !!
00021 //
00022 //
00023 //*************************************************************************
00024 
00025 #ifndef __MYADOLCNLP_HPP__
00026 #define __MYADOLCNLP_HPP__
00027 
00028 #include "IpTNLP.hpp"
00029 #include <adolc.h>
00030 
00031 #define tag_f 1
00032 #define tag_g 2
00033 #define tag_L 3
00034 
00035 using namespace Ipopt;
00036 
00037 class MyADOLC_NLP : public TNLP
00038 {
00039 public:
00040   /** default constructor */
00041   MyADOLC_NLP();
00042 
00043   /** default destructor */
00044   virtual ~MyADOLC_NLP();
00045 
00046   /**@name Overloaded from TNLP */
00047   //@{
00048   /** Method to return some info about the nlp */
00049   virtual bool get_nlp_info(Index& n, Index& m, Index& nnz_jac_g,
00050                             Index& nnz_h_lag, IndexStyleEnum& index_style);
00051 
00052   /** Method to return the bounds for my problem */
00053   virtual bool get_bounds_info(Index n, Number* x_l, Number* x_u,
00054                                Index m, Number* g_l, Number* g_u);
00055 
00056   /** Method to return the starting point for the algorithm */
00057   virtual bool get_starting_point(Index n, bool init_x, Number* x,
00058                                   bool init_z, Number* z_L, Number* z_U,
00059                                   Index m, bool init_lambda,
                                  Number* lambda);

  /** Template to return the objective value */
  template<class T> bool eval_obj(Index n, const T *x, T& obj_value);


  /** Template to compute contraints */
  template<class T> bool eval_constraints(Index n, const T *x, Index m, T *g);

  /** Original method from Ipopt to return the objective value */
  /** remains unchanged */
  virtual bool eval_f(Index n, const Number* x, bool new_x, Number& obj_value);

  /** Original method from Ipopt to return the gradient of the objective */
  /** remains unchanged */
  virtual bool eval_grad_f(Index n, const Number* x, bool new_x, Number* grad_f);

  /**  Original method from Ipopt to return the constraint residuals */
  /** remains unchanged */
  virtual bool eval_g(Index n, const Number* x, bool new_x, Index m, Number* g);

  /** Original method from Ipopt to return:
   *   1) The structure of the jacobian (if "values" is NULL)
   *   2) The values of the jacobian (if "values" is not NULL)
   */
  /** remains unchanged */
  virtual bool eval_jac_g(Index n, const Number* x, bool new_x,
                          Index m, Index nele_jac, Index* iRow, Index *jCol,
                          Number* values);

  /** Original method from Ipopt to return:
   *   1) The structure of the hessian of the lagrangian (if "values" is NULL)
   *   2) The values of the hessian of the lagrangian (if "values" is not NULL)
   */
  /** remains unchanged */
  virtual bool eval_h(Index n, const Number* x, bool new_x,
                      Number obj_factor, Index m, const Number* lambda,
                      bool new_lambda, Index nele_hess, Index* iRow,
                      Index* jCol, Number* values);

  //@}

  /** @name Solution Methods */
  //@{
  /** This method is called when the algorithm is complete so the TNLP can store/write the solution */
  virtual void finalize_solution(SolverReturn status,
                                 Index n, const Number* x, const Number* z_L, const Number* z_U,
                                 Index m, const Number* g, const Number* lambda,
                                 Number obj_value,
         const IpoptData* ip_data,
         IpoptCalculatedQuantities* ip_cq);
  //@}

//***************    start ADOL-C part ***********************************

 /** Method to generate the required tapes */
  virtual void generate_tapes(Index n, Index m);

//***************    end   ADOL-C part ***********************************

private:
  /**@name Methods to block default compiler methods.
   * The compiler automatically generates the following three methods.
   *  Since the default compiler implementation is generally not what
   *  you want (for all but the most simple classes), we usually
   *  put the declarations of these methods in the private section
   *  and never implement them. This prevents the compiler from
   *  implementing an incorrect "default" behavior without us
   *  knowing. (See Scott Meyers book, "Effective C++")
   *
   */
  //@{
  //  MyADOLC_NLP();
  MyADOLC_NLP(const MyADOLC_NLP&);
  MyADOLC_NLP& operator=(const MyADOLC_NLP&);
  //@}

  //@{
  double **Jac;

  double *x_lam;
  double **Hess;
  //@}

};

#endif
```

5.39 /home/lobianco/git/ffsm_pp/src/anyoption.cpp File Reference

```
#include "anyoption.h"
#include <cstring>
```

Include dependency graph for anyoption.cpp:



5.40 anyoption.cpp

```
/*
 * AnyOption 1.3
 *
 * kishan at hackorama dot com  www.hackorama.com JULY 2001
 *
 * + Acts as a common facade class for reading
 *   commandline options as well as options from
 *   an optionfile with delimited type value pairs
 *
 * + Handles the POSIX style single character options ( -w )
 *   as well as the newer GNU long options ( --width )
 *
 * + The option file assumes the traditional format of
 *   first character based comment lines and type value
 *   pairs with a delimiter , and flags which are not pairs
 *
 *    # this is a coment
 *    # next line is an option value pair
 *    width : 100
 *       # next line is a flag
 *      noimages
 *
 * + Supports printing out Help and Usage
 *
 * + Why not just use getopt() ?
 *
 *   getopt() Its a POSIX standard not part of ANSI-C.
 *   So it may not be available on platforms like Windows.
 *
 * + Why it is so long ?
 *
 *   The actual code which does command line parsing
 *   and option file parsing are done in  few methods.
 *   Most of the extra code are for providing a flexible
 *   common public interface to both a resourcefile and
 *   and command line supporting POSIX style and
 *   GNU long option as well as mixing of both.
 *
 * + Please see "anyoption.h" for public method descriptions
 *
 */
```

```

/* Updated Auguest 2004
 * Fix from  Michael D Peters (mpeters at sandia.gov)
 * to remove static local variables, allowing multiple instantiations
 * of the reader (for using multiple configuration files).  There is
 * an error in the destructor when using multiple instances, so you
 * cannot delete your objects (it will crash), but not calling the
 * destructor only introduces a small memory leak, so I
 * have not bothered tracking it down.
 *
 * Also updated to use modern C++ style headers, rather than
 * depricated iostream.h (it was causing my compiler problems)
*/

/*
 * Updated September 2006
 * Fix from Boyan Asenov for a bug in mixing up option indexes
 * leading to exception when mixing different options types
 */

#include "anyoption.h"
#include <cstring>

AnyOption::AnyOption()
{
  init();
}

AnyOption::AnyOption(int maxopt)
{
  init( maxopt , maxopt );
}

AnyOption::AnyOption(int maxopt, int maxcharopt)
{
  init( maxopt , maxcharopt );
}

AnyOption::~AnyOption()
{
  if( mem_allocated )
    cleanup();
}

void
AnyOption::init()
{
  init( DEFAULT_MAXOPTS , DEFAULT_MAXOPTS );
}

void
AnyOption::init(int maxopt, int maxcharopt )
{

  max_options    = maxopt;
  max_char_options = maxcharopt;
  max_usage_lines  = DEFAULT_MAXUSAGE;
  usage_lines  = 0 ;
  argc     = 0;
  argv     = NULL;
  posix_style  = true;
  verbose    = false;
  filename    = NULL;
  appname    = NULL;
  option_counter    = 0;
  optchar_counter  = 0;
  new_argv    = NULL;
  new_argc    = 0 ;
  max_legal_args    = 0 ;
  command_set    = false;
  file_set    = false;
  values      = NULL;
  g_value_counter = 0;
  mem_allocated    = false;
  command_set    = false;
  file_set  = false;
  opt_prefix_char     = '-';
  file_delimiter_char = ':';
  file_comment_char   = '#';
  equalsign   = '=';
  comment       = '#' ;
  delimiter     = ':' ;
  endofline     = '\n';
  whitespace    = ' ' ;
  nullterminate = '\0';
  set = false;
  once = true;
  hasoptions = false;
  autousage = false;

  strcpy( long_opt_prefix , "--" );

  if( alloc() == false ){
    cout << endl << "OPTIONS ERROR : Failed allocating memory" ;
    cout << endl ;
    cout << "Exiting." << endl ;
    exit (0);
  }
}

bool
AnyOption::alloc()
{
  int i = 0 ;
  int size = 0 ;

  if( mem_allocated )
    return true;

  size = (max_options+1) * sizeof(const char*);
  options = (const char**)malloc( size );
  optiontype = (int*) malloc( (max_options+1)*sizeof(int) );
  optionindex = (int*) malloc( (max_options+1)*sizeof(int) );
  if( options == NULL || optiontype == NULL || optionindex == NULL )
    return false;
  else
    mem_allocated  = true;
  for( i = 0 ; i < max_options ; i++ ){
    options[i] = NULL;
    optiontype[i] = 0 ;
    optionindex[i] = -1 ;
  }
  optionchars = (char*) malloc( (max_char_options+1)*sizeof(char) );
  optchartype = (int*) malloc( (max_char_options+1)*sizeof(int) );
  optcharindex = (int*) malloc( (max_char_options+1)*sizeof(int) );
  if( optionchars == NULL ||
            optchartype == NULL ||
            optcharindex == NULL )
        {
    mem_allocated = false;
    return false;
  }
  for( i = 0 ; i < max_char_options ; i++ ){
    optionchars[i] = '0';
    optchartype[i] = 0 ;
    optcharindex[i] = -1 ;
  }

  size = (max_usage_lines+1) * sizeof(const char*) ;
  usage = (const char**) malloc( size );

  if( usage == NULL  ){
    mem_allocated = false;
    return false;
  }
  for( i = 0 ; i < max_usage_lines ; i++ )
    usage[i] = NULL;

  return true;
}

bool
AnyOption::doubleOptStorage()
{
  options = (const char**)realloc( options,
      ((2*max_options)+1) * sizeof( const char*) );
  optiontype = (int*) realloc(  optiontype ,
      ((2 * max_options)+1)* sizeof(int) );
  optionindex = (int*) realloc(  optionindex,
      ((2 * max_options)+1) * sizeof(int) );
  if( options == NULL || optiontype == NULL || optionindex == NULL )
    return false;
  /* init new storage */
  for( int i = max_options ; i < 2*max_options ; i++ ){
    options[i] = NULL;
    optiontype[i] = 0 ;
    optionindex[i] = -1 ;
  }
  max_options = 2 * max_options ;
  return true;
}

bool
AnyOption::doubleCharStorage()
```

```
{
  optionchars = (char*) realloc( optionchars,
      ((2*max_char_options)+1)*sizeof(char) );
  optchartype = (int*) realloc( optchartype,
      ((2*max_char_options)+1)*sizeof(int) );
  optcharindex = (int*) realloc( optcharindex,
      ((2*max_char_options)+1)*sizeof(int) );
  if( optionchars == NULL ||
      optchartype == NULL ||
      optcharindex == NULL )
    return false;
  /* init new storage */
  for( int i = max_char_options ; i < 2*max_char_options ; i++ ){
    optionchars[i] = ’0’;
    optchartype[i] = 0 ;
    optcharindex[i] = -1 ;
  }
  max_char_options = 2 * max_char_options;
  return true;
}

bool
AnyOption::doubleUsageStorage()
{
  usage = (const char**)realloc( usage,
      ((2*max_usage_lines)+1) * sizeof( const char*) );
  if ( usage == NULL )
    return false;
  for( int i = max_usage_lines ; i < 2*max_usage_lines ; i++ )
    usage[i] = NULL;
  max_usage_lines = 2 * max_usage_lines ;
  return true;

}


void
AnyOption::cleanup()
{
  free (options);
  free (optiontype);
  free (optionindex);
  free (optionchars);
  free (optchartype);
  free (optcharindex);
  free (usage);
  if( values != NULL )
    free (values);
  if( new_argv != NULL )
    free (new_argv);
}

void
AnyOption::setCommandPrefixChar( char _prefix )
{
  opt_prefix_char = _prefix;
}

void
AnyOption::setCommandLongPrefix( char *_prefix )
{
  if( strlen( _prefix ) > MAX_LONG_PREFIX_LENGTH ){
    *( _prefix + MAX_LONG_PREFIX_LENGTH ) = ’\0’;
  }

  strcpy (long_opt_prefix,  _prefix);
}

void
AnyOption::setFileCommentChar( char _comment )
{
  file_delimiter_char = _comment;
}


void
AnyOption::setFileDelimiterChar( char _delimiter )
{
  file_comment_char = _delimiter ;
}

bool
AnyOption::CommandSet()
{
  return( command_set );
}

```

```
bool
AnyOption::FileSet()
{
  return( file_set );
}

void
AnyOption::noPOSIX()
{
  posix_style = false;
}

bool
AnyOption::POSIX()
{
  return posix_style;
}


void
AnyOption::setVerbose()
{
  verbose = true ;
}

void
AnyOption::printVerbose()
{
  if( verbose )
    cout << endl  ;
}
void
AnyOption::printVerbose( const char *msg )
{
  if( verbose )
    cout << msg  ;
}

void
AnyOption::printVerbose( char *msg )
{
  if( verbose )
    cout << msg  ;
}

void
AnyOption::printVerbose( char ch )
{
  if( verbose )
    cout << ch ;
}

bool
AnyOption::hasOptions()
{
  return hasoptions;
}

void
AnyOption::autoUsagePrint(bool _autousage)
{
  autousage = _autousage;
}

void
AnyOption::useCommandArgs( int _argc, char **_argv )
{
  argc = _argc;
  argv = _argv;
  command_set = true;
  appname = argv[0];
  if(argc > 1) hasoptions = true;
}

void
AnyOption::useFiileName( const char *_filename )
{
  filename = _filename;
  file_set = true;
}

/*
 * set methods for options
 */

void
AnyOption::setCommandOption( const char *opt )
```

```
{
  addOption( opt , COMMAND_OPT );
  g_value_counter++;
}

void
AnyOption::setCommandOption( char opt )
{
  addOption( opt , COMMAND_OPT );
  g_value_counter++;
}

void
AnyOption::setCommandOption( const char *opt , char optchar )
{
  addOption( opt , COMMAND_OPT );
  addOption( optchar , COMMAND_OPT );
  g_value_counter++;
}

void
AnyOption::setCommandFlag( const char *opt )
{
  addOption( opt , COMMAND_FLAG );
  g_value_counter++;
}

void
AnyOption::setCommandFlag( char opt )
{
  addOption( opt , COMMAND_FLAG );
  g_value_counter++;
}

void
AnyOption::setCommandFlag( const char *opt , char optchar )
{
  addOption( opt , COMMAND_FLAG );
  addOption( optchar , COMMAND_FLAG );
  g_value_counter++;
}

void
AnyOption::setFileOption( const char *opt )
{
  addOption( opt , FILE_OPT );
  g_value_counter++;
}

void
AnyOption::setFileOption( char opt )
{
  addOption( opt , FILE_OPT );
  g_value_counter++;
}

void
AnyOption::setFileOption( const char *opt , char optchar )
{
  addOption( opt , FILE_OPT );
  addOption( optchar, FILE_OPT  );
  g_value_counter++;
}

void
AnyOption::setFileFlag( const char *opt )
{
  addOption( opt , FILE_FLAG );
  g_value_counter++;
}

void
AnyOption::setFileFlag( char opt )
{
  addOption( opt , FILE_FLAG );
  g_value_counter++;
}

void
AnyOption::setFileFlag( const char *opt , char optchar )
{
  addOption( opt , FILE_FLAG );
  addOption( optchar , FILE_FLAG );
  g_value_counter++;
}

void
```

```
AnyOption::setOption( const char *opt )
{
  addOption( opt , COMMON_OPT );
  g_value_counter++;
}

void
AnyOption::setOption( char opt )
{
  addOption( opt , COMMON_OPT );
  g_value_counter++;
}

void
AnyOption::setOption( const char *opt , char optchar )
{
  addOption( opt , COMMON_OPT );
  addOption( optchar , COMMON_OPT );
  g_value_counter++;
}

void
AnyOption::setFlag( const char *opt )
{
  addOption( opt , COMMON_FLAG );
  g_value_counter++;
}

void
AnyOption::setFlag( const char opt )
{
  addOption( opt , COMMON_FLAG );
  g_value_counter++;
}

void
AnyOption::setFlag( const char *opt , char optchar )
{
  addOption( opt , COMMON_FLAG );
  addOption( optchar , COMMON_FLAG );
  g_value_counter++;
}

void
AnyOption::addOption( const char *opt, int type )
{
  if( option_counter >= max_options ){
    if( doubleOptStorage() == false ){
      addOptionError( opt );
      return;
    }
  }
  options[ option_counter ] = opt ;
  optiontype[ option_counter ] =  type ;
  optionindex[ option_counter ] = g_value_counter;
  option_counter++;
}

void
AnyOption::addOption( char opt, int type )
{
  if( !POSIX() ){
    printVerbose("Ignoring the option character \"");
    printVerbose(  opt );
    printVerbose( "\" ( POSIX options are turned off )" );
    printVerbose();
    return;
  }


  if( optchar_counter >= max_char_options ){
    if( doubleCharStorage() == false ){
      addOptionError( opt );
      return;
    }
  }
  optionchars[ optchar_counter ] =  opt ;
  optchartype[ optchar_counter ] =  type ;
  optcharindex[ optchar_counter ] = g_value_counter;
  optchar_counter++;
}

void
AnyOption::addOptionError( const char *opt )
{
  cout << endl ;
  cout << "OPTIONS ERROR : Failed allocating extra memory " << endl ;
  cout << "While adding the option : \""<< opt << "\"" << endl;
  cout << "Exiting." << endl ;
  cout << endl ;
  exit(0);
}

void
AnyOption::addOptionError( char opt )
{
  cout << endl ;
  cout << "OPTIONS ERROR : Failed allocating extra memory " << endl ;
  cout << "While adding the option: \""<< opt << "\"" << endl;
  cout << "Exiting." << endl ;
  cout << endl ;
  exit(0);
}

void
AnyOption::processOptions()
{
  if( ! valueStoreOK() )
    return;
}

void
AnyOption::processCommandArgs(int max_args)
{
  max_legal_args = max_args;
  processCommandArgs();
}

void
AnyOption::processCommandArgs( int _argc, char **_argv, int max_args )
{
  max_legal_args = max_args;
  processCommandArgs(  _argc, _argv );
}

void
AnyOption::processCommandArgs( int _argc, char **_argv )
{
  useCommandArgs( _argc, _argv );
  processCommandArgs();
}

void
AnyOption::processCommandArgs()
{
     if( ! ( valueStoreOK() && CommandSet() )  )
     return;

  if( max_legal_args == 0 )
    max_legal_args = argc;
  new_argv = (int*) malloc( (max_legal_args+1) * sizeof(int) );
  for( int i = 1 ; i < argc ; i++ ){/* ignore first argv */
    if(  argv[i][0] == long_opt_prefix[0] &&
                     argv[i][1] == long_opt_prefix[1] ) { /* long GNU option */
      int match_at = parseGNU( argv[i]+2 ); /* skip -- */
      if( match_at >= 0 && i < argc-1 ) /* found match */
        setValue( options[match_at] , argv[++i] );
    }else if(  argv[i][0] ==  opt_prefix_char ) { /* POSIX char */
      if( POSIX() ){
        char ch =  parsePOSIX( argv[i]+1 );/* skip - */
        if( ch != '0' && i < argc-1 ) /* matching char */
          setValue( ch ,  argv[++i] );
      } else { /* treat it as GNU option with a - */
        int match_at = parseGNU( argv[i]+1 ); /* skip - */
        if( match_at >= 0 && i < argc-1 ) /* found match */
          setValue( options[match_at] , argv[++i] );
      }
    }else { /* not option but an argument keep index */
      if( new_argc < max_legal_args ){
                                new_argv[ new_argc ] = i ;
                                new_argc++;
                        }else{ /* ignore extra arguments */
                                printVerbose( "Ignoring extra argument: " );
        printVerbose( argv[i] );
        printVerbose( );
        printAutoUsage();
                        }
      printVerbose( "Unknown command argument option : " );
      printVerbose( argv[i] );
      printVerbose( );
      printAutoUsage();
    }
  }
}
```

```

char
AnyOption::parsePOSIX( char* arg )
{

  for( unsigned int i = 0 ; i < strlen(arg) ; i++ ){
    char ch = arg[i] ;
    if( matchChar(ch) ) { /* keep matching flags till an option */
      /*if last char argv[++i] is the value */
      if( i == strlen(arg)-1 ){
        return ch;
      }else{/* else the rest of arg is the value */
        i++; /* skip any ’=’ and ’ ’ */
        while( arg[i] == whitespace
              || arg[i] == equalsign )
          i++;
        setValue( ch , arg+i );
        return ’0’;
      }
    }
  }
  printVerbose( "Unknown command argument option : " );
  printVerbose( arg );
  printVerbose( );
  printAutoUsage();
  return ’0’;
}

int
AnyOption::parseGNU( char *arg )
{
  int split_at = 0;
  /* if has a ’=’ sign get value */
  for( unsigned int i = 0 ; i < strlen(arg) ; i++ ){
    if(arg[i] ==  equalsign ){
      split_at = i ; /* store index */
      i = strlen(arg); /* get out of loop */
    }
  }
  if( split_at > 0 ){ /* it is an option value pair */
    char* tmp = (char*) malloc(  (split_at+1)*sizeof(char) );
    for( int i = 0 ; i < split_at ; i++ )
      tmp[i] = arg[i];
    tmp[split_at] = ’\0’;

    if ( matchOpt( tmp ) >= 0 ){
      setValue( options[matchOpt(tmp)] , arg+split_at+1 );
      free (tmp);
    }else{
      printVerbose( "Unknown command argument option : " );
      printVerbose( arg );
      printVerbose( );
      printAutoUsage();
      free (tmp);
      return -1;
    }
  }else{ /* regular options with no ’=’ sign  */
    return  matchOpt(arg);
  }
  return -1;
}


int
AnyOption::matchOpt( char *opt )
{
  for( int i = 0 ; i < option_counter ; i++ ){
    if( strcmp( options[i], opt ) == 0 ){
      if( optiontype[i] ==  COMMON_OPT ||
          optiontype[i] ==  COMMAND_OPT )
      { /* found option return index */
        return i;
      }else if( optiontype[i] == COMMON_FLAG ||
             optiontype[i] == COMMAND_FLAG )
      { /* found flag, set it */
        setFlagOn( opt );
        return -1;
      }
    }
  }
  printVerbose( "Unknown command argument option : " );
  printVerbose( opt  ) ;
  printVerbose( );
  printAutoUsage();
  return  -1;
}
bool
```

```
AnyOption::matchChar( char c )
{
  for( int i = 0 ; i < optchar_counter ; i++ ){
    if( optionchars[i] == c ) { /* found match */
      if(optchartype[i] == COMMON_OPT ||
           optchartype[i] == COMMAND_OPT )
      { /* an option store and stop scanning */
        return true;
      }else if( optchartype[i] == COMMON_FLAG ||
          optchartype[i] == COMMAND_FLAG ) { /* a flag store and keep scanning */
        setFlagOn( c );
        return false;
      }
    }
  }
  printVerbose( "Unknown command argument option : " );
  printVerbose( c ) ;
  printVerbose( );
  printAutoUsage();
  return false;
}

bool
AnyOption::valueStoreOK( )
{
  int size= 0;
  if( !set ){
    if( g_value_counter > 0 ){
      size = g_value_counter * sizeof(char*);
      values = (char**)malloc( size );
      for( int i = 0 ; i < g_value_counter ; i++)
        values[i] = NULL;
      set = true;
    }
  }
  return  set;
}

/*
 * public get methods
 */
char*
AnyOption::getValue( const char *option )
{
  if( !valueStoreOK() )
    return NULL;

  for( int i = 0 ; i < option_counter ; i++ ){
    if( strcmp( options[i], option ) == 0 )
      return values[ optionindex[i] ];
  }
  return NULL;
}

bool
AnyOption::getFlag( const char *option )
{
  if( !valueStoreOK() )
    return false;
  for( int i = 0 ; i < option_counter ; i++ ){
    if( strcmp( options[i], option ) == 0 )
      return findFlag( values[ optionindex[i] ] );
  }
  return false;
}

char*
AnyOption::getValue( char option )
{
  if( !valueStoreOK() )
    return NULL;
  for( int i = 0 ; i < optchar_counter ; i++ ){
    if( optionchars[i] == option )
      return values[ optcharindex[i] ];
  }
  return NULL;
}

bool
AnyOption::getFlag( char option )
{
  if( !valueStoreOK() )
    return false;
  for( int i = 0 ; i < optchar_counter ; i++ ){
    if( optionchars[i] == option )
      return findFlag( values[ optcharindex[i] ] ) ;
  }
  return false;
}

bool
AnyOption::findFlag( char* val )
{
  if( val == NULL )
    return false;

  if( strcmp( TRUE_FLAG , val ) == 0 )
    return true;

  return false;
}

/*
 * private set methods
 */
bool
AnyOption::setValue( const char *option , char *value )
{
  if( !valueStoreOK() )
    return false;
        for( int i = 0 ; i < option_counter ; i++ ){
                if( strcmp( options[i], option ) == 0 ){
                        values[ optionindex[i] ] = (char*) malloc((strlen(value)+1)*sizeof(char));
                        strcpy( values[ optionindex[i] ], value );
      return true;
    }
        }
        return false;
}

bool
AnyOption::setFlagOn( const char *option )
{
  if( !valueStoreOK() )
    return false;
        for( int i = 0 ; i < option_counter ; i++ ){
                if( strcmp( options[i], option ) == 0 ){
                        values[ optionindex[i] ] = (char*) malloc((strlen(TRUE_FLAG)+1)*sizeof(char));
                        strcpy( values[ optionindex[i] ]  ,  TRUE_FLAG );
      return true;
    }
        }
        return false;
}

bool
AnyOption::setValue( char option , char *value )
{
  if( !valueStoreOK() )
    return false;
        for( int i = 0 ; i < optchar_counter ; i++ ){
                if( optionchars[i] == option ){
                        values[ optcharindex[i] ] = (char*) malloc((strlen(value)+1)*sizeof(char));
                        strcpy( values[ optcharindex[i] ],  value );
      return true;
    }
        }
        return false;
}

bool
AnyOption::setFlagOn( char option )
{
  if( !valueStoreOK() )
    return false;
        for( int i = 0 ; i < optchar_counter ; i++ ){
                if( optionchars[i] == option ){
                        values[ optcharindex[i] ] = (char*) malloc((strlen(TRUE_FLAG)+1)*sizeof(char));
      strcpy( values[ optcharindex[i] ] , TRUE_FLAG );
      return true;
    }
        }
        return false;
}


int
AnyOption::getArgc( )
{
  return new_argc;
```

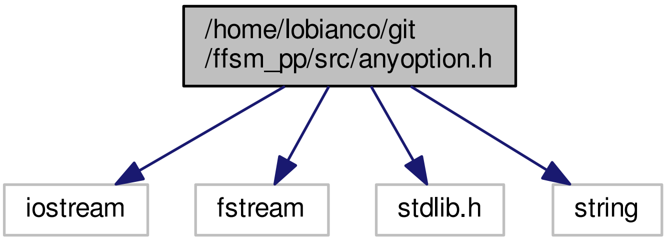
```
}

char*
AnyOption::getArgv( int index )
{
  if( index < new_argc ){
    return ( argv[ new_argv[ index ] ] );
  }
  return NULL;
}

/* dotfile sub routines */

bool
AnyOption::processFile()
{
  if( ! (valueStoreOK() && FileSet())  )
    return false;
  return  ( consumeFile(readFile()) );
}

bool
AnyOption::processFile( const char *filename )
{
  useFiileName(filename );
  return ( processFile() );
}

char*
AnyOption::readFile()
{
  return ( readFile(filename) );
}

/*
 * read the file contents to a character buffer
 */

char*
AnyOption::readFile( const char* fname )
{
        int length;
        char *buffer;
        ifstream is;
        is.open ( fname , ifstream::in );
        if( ! is.good() ){
                is.close();
                return NULL;
        }
        is.seekg (0, ios::end);
        length = is.tellg();
        is.seekg (0, ios::beg);
        buffer = (char*) malloc(length*sizeof(char));
        is.read (buffer,length);
        is.close();
        return buffer;
}

/*
 * scans a char* buffer for lines that does not
 * start with the specified comment character.
 */
bool
AnyOption::consumeFile( char *buffer )
{

        if( buffer == NULL )
    return false;

         char *cursor = buffer;/* preserve the ptr */
         char *pline = NULL ;
         int linelength = 0;
         bool newline = true;
         for( unsigned int i = 0 ; i < strlen( buffer ) ; i++ ){
         if( *cursor == endofline ) { /* end of line */
            if( pline != NULL ) /* valid line */
                   processLine( pline, linelength );
                   pline = NULL;
                   newline = true;
             }else if( newline ){ /* start of line */
                   newline = false;
                  if( (*cursor != comment ) ){ /* not a comment */
            pline = cursor ;
                        linelength = 0 ;
                  }
               }
              cursor++; /* keep moving */
              linelength++;
         }
       free (buffer);
  return true;
}


/*
 *  find a valid type value pair separated by a delimiter
 *  character and pass it to valuePairs()
 *  any line which is not valid will be considered a value
 *  and will get passed on to justValue()
 *
 *  assuming delimiter is ':' the behaviour will be,
 *
 *  width:10    - valid pair valuePairs( width, 10 );
 *  width : 10  - valid pair valuepairs( width, 10 );
 *
 *  ::::        - not valid
 *  width       - not valid
 *  :10         - not valid
 *  width:      - not valid
 *  ::          - not valid
 *  :           - not valid
 *
 */

void
AnyOption::processLine( char *theline, int length  )
{
        bool found = false;
        char *pline = (char*) malloc( (length+1)*sizeof(char) );
        for( int i = 0 ; i < length ; i ++ )
                pline[i]= *(theline++);
        pline[length] = nullterminate;
        char *cursor = pline ; /* preserve the ptr */
        if( *cursor == delimiter || *(cursor+length-1) == delimiter ){
                justValue( pline );/* line with start/end delimiter */
        }else{
                for( int i = 1 ; i < length-1 && !found ; i++){/* delimiter */
                        if( *cursor == delimiter ){
                                *(cursor-1) = nullterminate; /* two strings */
                                found = true;
                                valuePairs( pline , cursor+1 );
                        }
                        cursor++;
                }
                cursor++;
                if( !found ) /* not a pair */
                        justValue( pline );
        }
        free (pline);
}

/*
 * removes trailing and preceeding whitespaces from a string
 */
char*
AnyOption::chomp( char *str )
{
        while( *str == whitespace )
                str++;
        char *end = str+strlen(str)-1;
        while( *end == whitespace )
                end--;
        *(end+1) = nullterminate;
        return str;
}

void
AnyOption::valuePairs( char *type, char *value )
{
  if ( strlen(chomp(type)) == 1  ){ /* this is a char option */
    for( int i = 0 ; i < optchar_counter ; i++ ){
      if(  optionchars[i] == type[0]  ){ /* match */
        if( optchartype[i] == COMMON_OPT ||
            optchartype[i] == FILE_OPT )
        {
           setValue( type[0] , chomp(value) );
           return;
        }
      }
    }
  }
  /* if no char options matched */
  for( int i = 0 ; i < option_counter ; i++ ){
    if( strcmp( options[i], type ) == 0 ){ /* match */
```

```
      if( optiontype[i] == COMMON_OPT ||
          optiontype[i] == FILE_OPT )
      {
         setValue( type , chomp(value) );
         return;
      }
    }
  }
        printVerbose( "Unknown option in resourcefile : " );
  printVerbose( type );
  printVerbose( );
}

void
AnyOption::justValue( char *type )
{

  if ( strlen(chomp(type)) == 1  ){ /* this is a char option */
    for( int i = 0 ; i < optchar_counter ; i++ ){
      if(  optionchars[i] == type[0]  ){ /* match */
        if( optchartype[i] == COMMON_FLAG ||
            optchartype[i] == FILE_FLAG )
        {
           setFlagOn( type[0] );
           return;
        }
      }
    }
  }
  /* if no char options matched */
  for( int i = 0 ; i < option_counter ; i++ ){
    if( strcmp( options[i], type ) == 0 ){ /* match */
      if( optiontype[i] == COMMON_FLAG ||
          optiontype[i] == FILE_FLAG )
      {
         setFlagOn( type );
         return;
      }
    }
  }
        printVerbose( "Unknown option in resourcefile : " );
  printVerbose( type  );
  printVerbose( );
}

/*
 * usage and help
 */


void
AnyOption::printAutoUsage()
{
  if( autousage ) printUsage();
}

void
AnyOption::printUsage()
{

  if( once ) {
    once = false ;
    cout << endl ;
    for( int i = 0 ; i < usage_lines ; i++ )
      cout << usage[i] << endl ;
    cout << endl ;
  }
}


void
AnyOption::addUsage( const char *line )
{
  if( usage_lines >= max_usage_lines ){
    if( doubleUsageStorage() == false ){
      addUsageError( line );
      exit(1);
    }
  }
  usage[ usage_lines ] = line ;
  usage_lines++;
}

void
AnyOption::addUsageError( const char *line )
{
  cout << endl ;
  cout << "OPTIONS ERROR : Failed allocating extra memory " << endl ;
  cout << "While adding the usage/help  : \""<< line << "\"" << endl;
  cout << "Exiting." << endl ;
  cout << endl ;
  exit(0);

}
```

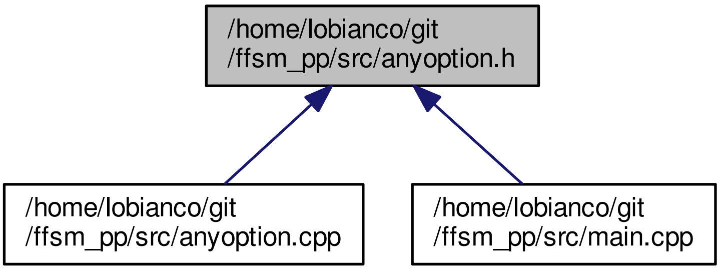
5.41 /home/lobianco/git/ffsm_pp/src/anyoption.h File Reference

```
#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <string>
```

Include dependency graph for anyoption.h:



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



Classes

- class AnyOption

Macros

- #define COMMON_OPT 1
- #define COMMAND_OPT 2
- #define FILE_OPT 3
- #define COMMON_FLAG 4
- #define COMMAND_FLAG 5
- #define FILE_FLAG 6
- #define COMMAND_OPTION_TYPE 1
- #define COMMAND_FLAG_TYPE 2
- #define FILE_OPTION_TYPE 3
- #define FILE_FLAG_TYPE 4
- #define UNKNOWN_TYPE 5
- #define DEFAULT_MAXOPTS 10
- #define MAX_LONG_PREFIX_LENGTH 2
- #define DEFAULT_MAXUSAGE 3
- #define DEFAULT_MAXHELP 10
- #define TRUE_FLAG "true"

5.41.1 Macro Definition Documentation

5.41.1.1 #define COMMAND_FLAG 5

Definition at line 13 of file anyoption.h.

Referenced by AnyOption::matchChar(), AnyOption::matchOpt(), and AnyOption::setCommandFlag().

5.41.1.2 #define COMMAND_FLAG_TYPE 2

Definition at line 17 of file anyoption.h.

5.41.1.3 #define COMMAND_OPT 2

Definition at line 10 of file anyoption.h.

Referenced by AnyOption::matchChar(), AnyOption::matchOpt(), and AnyOption::setCommandOption().

5.41.1.4 #define COMMAND_OPTION_TYPE 1

Definition at line 16 of file anyoption.h.

5.41.1.5 #define COMMON_FLAG 4

Definition at line 12 of file anyoption.h.

Referenced by AnyOption::justValue(), AnyOption::matchChar(), AnyOption::matchOpt(), and AnyOption::setFlag().

5.41.1.6 #define COMMON_OPT 1

Definition at line 9 of file anyoption.h.

Referenced by AnyOption::matchChar(), AnyOption::matchOpt(), AnyOption::setOption(), and AnyOption::value↩Pairs().

5.41.1.7 #define DEFAULT_MAXHELP 10

Definition at line 26 of file anyoption.h.

5.41.1.8 #define DEFAULT_MAXOPTS 10

Definition at line 22 of file anyoption.h.

Referenced by AnyOption::init().

5.41.1.9 #define DEFAULT_MAXUSAGE 3

Definition at line 25 of file anyoption.h.

Referenced by AnyOption::init().

5.41.1.10 #define FILE_FLAG 6

Definition at line 14 of file anyoption.h.

Referenced by AnyOption::justValue(), and AnyOption::setFileFlag().

5.41.1.11 #define FILE_FLAG_TYPE 4

Definition at line 19 of file anyoption.h.

5.41.1.12 #define FILE_OPT 3

Definition at line 11 of file anyoption.h.

Referenced by AnyOption::setFileOption(), and AnyOption::valuePairs().

5.41.1.13 #define FILE_OPTION_TYPE 3

Definition at line 18 of file anyoption.h.

5.41.1.14 #define MAX_LONG_PREFIX_LENGTH 2

Definition at line 23 of file anyoption.h.

Referenced by AnyOption::setCommandLongPrefix().

5.41.1.15 #define TRUE_FLAG "true"

Definition at line 28 of file anyoption.h.

Referenced by AnyOption::findFlag(), and AnyOption::setFlagOn().

5.41.1.16 #define UNKNOWN_TYPE 5

Definition at line 20 of file anyoption.h.

5.42 anyoption.h

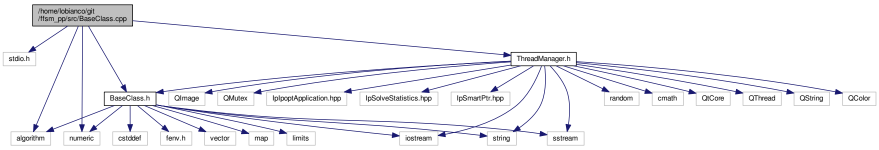
```
#ifndef _ANYOPTION_H
#define _ANYOPTION_H

#include <iostream>
#include <fstream>
#include <stdlib.h>
#include <string>

#define COMMON_OPT   1
#define COMMAND_OPT   2
#define FILE_OPT   3
#define COMMON_FLAG   4
#define COMMAND_FLAG   5
#define FILE_FLAG   6

#define COMMAND_OPTION_TYPE    1
#define COMMAND_FLAG_TYPE   2
#define FILE_OPTION_TYPE    3
#define FILE_FLAG_TYPE     4
#define UNKNOWN_TYPE     5

#define DEFAULT_MAXOPTS   10
#define MAX_LONG_PREFIX_LENGTH   2

#define DEFAULT_MAXUSAGE  3
#define DEFAULT_MAXHELP         10

#define TRUE_FLAG "true"

using namespace std;

class AnyOption
{

public: /* the public interface */
  AnyOption();
  AnyOption(int maxoptions );
  AnyOption(int maxoptions , int maxcharoptions);
  ~AnyOption();

  /*
         * following set methods specifies the
   * special characters and delimiters
   * if not set traditional defaults will be used
         */

  void setCommandPrefixChar( char _prefix );   /* ’-’ in "-w" */
  void setCommandLongPrefix( char *_prefix );  /* ’--’ in "--width" */
  void setFileCommentChar( char _comment );    /* ’#’ in shellscripts */
  void setFileDelimiterChar( char _delimiter );/* ’:’ in "width : 100" */

  /*
   * provide the input for the options
         * like argv[] for commndline and the
         * option file name  to use;
   */

  void useCommandArgs( int _argc, char **_argv );
  void useFiileName( const char *_filename );

  /*
         * turn off the POSIX style options
         * this means anything starting with a ’-’ or "--"
         * will be considered a valid option
         * which alo means you cannot add a bunch of
         * POIX options chars together like "-lr"  for "-l -r"
         *
         */

  void noPOSIX();

  /*
         * prints warning verbose if you set anything wrong
         */
  void setVerbose();


  /*
         * there are two types of options
         *
         * Option - has an associated value ( -w 100 )
         * Flag  - no value, just a boolean flag  ( -nogui )
         *
   * the options can be either a string ( GNU style )
```

```
         * or a character ( traditional POSIX style )
         * or both ( --width, -w )
         *
         * the options can be common to the commandline and
         * the optionfile, or can belong only to either of
         * commandline and optionfile
         *
         * following set methods, handle all the aboove
   * cases of options.
         */

  /* options comman to command line and option file */
  void setOption( const char *opt_string );
  void setOption( char  opt_char );
  void setOption( const char *opt_string , char opt_char );
  void setFlag( const char *opt_string );
  void setFlag( char  opt_char );
  void setFlag( const char *opt_string , char opt_char );

  /* options read from commandline only */
  void setCommandOption( const char *opt_string );
  void setCommandOption( char  opt_char );
  void setCommandOption( const char *opt_string , char opt_char );
  void setCommandFlag( const char *opt_string );
  void setCommandFlag( char  opt_char );
  void setCommandFlag( const char *opt_string , char opt_char );

  /* options read from an option file only  */
  void setFileOption( const char *opt_string );
  void setFileOption( char  opt_char );
  void setFileOption( const char *opt_string , char opt_char );
  void setFileFlag( const char *opt_string );
  void setFileFlag( char  opt_char );
  void setFileFlag( const char *opt_string , char opt_char );

  /*
         * process the options, registerd using
         * useCommandArgs() and useFileName();
         */
  void processOptions();
  void processCommandArgs();
  void processCommandArgs( int max_args );
  bool processFile();

  /*
         * process the specified options
         */
  void processCommandArgs( int _argc, char **_argv );
  void processCommandArgs( int _argc, char **_argv, int max_args );
  bool processFile( const char *_filename );

  /*
         * get the value of the options
   * will return NULL if no value is set
         */
  char *getValue( const char *_option );
  bool  getFlag( const char *_option );
  char *getValue( char _optchar );
  bool  getFlag( char _optchar );

  /*
   * Print Usage
   */
  void printUsage();
  void printAutoUsage();
  void addUsage( const char *line );
  void printHelp();
        /* print auto usage printing for unknown options or flag */
  void autoUsagePrint(bool flag);

  /*
         * get the argument count and arguments sans the options
         */
  int   getArgc();
  char* getArgv( int index );
  bool  hasOptions();

private: /* the hidden data structure */
  int argc;    /* commandline arg count  */
  char **argv;      /* commndline args */
  const char* filename;   /* the option file */
  char* appname;   /* the application name from argv[0] */

  int *new_argv;     /* arguments sans options (index to argv) */
  int new_argc;     /* argument count sans the options */
  int max_legal_args;   /* ignore extra arguments */


  /* option strings storage + indexing */
  int max_options;   /* maximum number of options */
  const char **options;   /* storage */
  int *optiontype;   /* type - common, command, file */
  int *optionindex;  /* index into value storage */
  int option_counter;   /* counter for added options  */

  /* option chars storage + indexing */
  int max_char_options;   /* maximum number options */
  char *optionchars;   /*  storage */
  int *optchartype;   /* type - common, command, file */
  int *optcharindex;   /* index into value storage */
  int optchar_counter;   /* counter for added options  */

  /* values */
  char **values;     /* common value storage */
  int g_value_counter;   /* globally updated value index LAME! */

  /* help and usage */
  const char **usage;   /* usage */
  int max_usage_lines;  /* max usage lines reseverd */
  int usage_lines;  /* number of usage lines */

  bool command_set;  /* if argc/argv were provided */
  bool file_set;    /* if a filename was provided */
  bool mem_allocated;     /* if memory allocated in init() */
  bool posix_style;   /* enables to turn off POSIX style options */
  bool verbose;    /* silent|verbose */
  bool print_usage;  /* usage verbose */
  bool print_help;  /* help verbose */

  char opt_prefix_char;    /*  '-' in "-w" */
  char long_opt_prefix[MAX_LONG_PREFIX_LENGTH + 1]; /* '--' in "--width" */
  char file_delimiter_char;  /* ':' in width : 100 */
  char file_comment_char;    /*  '#' in "#this is a comment" */
  char equalsign;
  char comment;
  char delimiter;
  char endofline;
  char whitespace;
  char nullterminate;

  bool set;   //was static member
  bool once;  //was static member

  bool hasoptions;
  bool autousage;

private: /* the hidden utils */
  void init();
  void init(int maxopt, int maxcharopt );
  bool alloc();
  void cleanup();
  bool valueStoreOK();

  /* grow storage arrays as required */
  bool doubleOptStorage();
  bool doubleCharStorage();
  bool doubleUsageStorage();

  bool setValue( const char *option , char *value );
  bool setFlagOn( const char *option );
  bool setValue( char optchar , char *value);
  bool setFlagOn( char optchar );

  void addOption( const char* option , int type );
  void addOption( char optchar , int type );
  void addOptionError( const char *opt);
  void addOptionError( char opt);
  bool findFlag( char* value );
  void addUsageError( const char *line );
  bool CommandSet();
  bool FileSet();
  bool POSIX();

  char parsePOSIX( char* arg );
  int parseGNU( char *arg );
  bool matchChar( char c );
  int matchOpt( char *opt );

  /* dot file methods */
  char *readFile();
  char *readFile( const char* fname );
  bool consumeFile( char *buffer );
  void processLine( char *theline, int length );
  char *chomp( char *str );
  void valuePairs( char *type, char *value );
  void justValue( char *value );

  void printVerbose( const char *msg );
  void printVerbose( char *msg );
  void printVerbose( char ch );
  void printVerbose( );


};

#endif /* ! _ANYOPTION_H */
```

5.43 /home/lobianco/git/ffsm_pp/src/BaseClass.cpp File Reference

```
#include <stdio.h>
#include <algorithm>
#include <numeric>
#include "BaseClass.h"
#include "ThreadManager.h"
```

Include dependency graph for BaseClass.cpp:



5.44 BaseClass.cpp

```
/****************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière                *
 *   http://ffsm-project.org                                                *
 *                                                                          *
 *   This program is free software; you can redistribute it and/or modify   *
 *   it under the terms of the GNU General Public License as published by   *
 *   the Free Software Foundation; either version 3 of the License, or      *
 *   (at your option) any later version, given the compliance with the      *
 *   exceptions listed in the file COPYING that is distribued together      *
 *   with this file.                                                        *
 *                                                                          *
 *   This program is distributed in the hope that it will be useful,        *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of         *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the          *
 *   GNU General Public License for more details.                           *
 *                                                                          *
 *   You should have received a copy of the GNU General Public License      *
 *   along with this program; if not, write to the                          *
 *   Free Software Foundation, Inc.,                                        *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.              *
 ****************************************************************************/
#include <stdio.h>
#include <algorithm>
#include <numeric>

#include "BaseClass.h"
#include "ThreadManager.h"

using namespace std;

BaseClass::BaseClass()
{
  MTHREAD=NULL;
}

BaseClass::~BaseClass()
{

}
```

```

/**
Overloaded method for the output log:

@param msgCode_h:    MSG_DEBUG, MSG_INFO, MSG_WARNING, MSG_ERROR, MSG_CRITICAL_ERROR
@param msg_h:        message text (string)
@param refreshGUI_h: use this call to "ping" the GUI (optional, default=true)

*/
void
BaseClass::msgOut(const int& msgCode_h, const string& msg_h, const bool& refreshGUI_h) const {

  msgOut2(msgCode_h, msg_h, refreshGUI_h);

}

/**
Overloaded method for the output log:

@param msgCode_h:    MSG_DEBUG, MSG_INFO, MSG_WARNING, MSG_ERROR, MSG_CRITICAL_ERROR
@param msg_h:        message text (int)
@param refreshGUI_h: use this call to "ping" the GUI (optional, default=true)

*/
void
BaseClass::msgOut(const int& msgCode_h, const int& msg_h, const bool& refreshGUI_h) const {
  msgOut2(msgCode_h, i2s(msg_h), refreshGUI_h);
}

/**
Overloaded method for the output log:

@param msgCode_h:    MSG_DEBUG, MSG_INFO, MSG_WARNING, MSG_ERROR, MSG_CRITICAL_ERROR
@param msg_h:        message text (double)
@param refreshGUI_h: use this call to "ping" the GUI (optional, default=true)

*/
void
BaseClass::msgOut(const int& msgCode_h, const double& msg_h, const bool& refreshGUI_h) const {
  msgOut2(msgCode_h, d2s(msg_h), refreshGUI_h);

}

/**
Convenient (private) function to actually do the job of the overloaded functions

*/
void
BaseClass::msgOut2(const int& msgCode_h, const string& msg_h, const bool& refreshGUI_h) const {

  string prefix;
  switch (msgCode_h){
  case MSG_NO_MSG:
    return;
  case MSG_DEBUG:
    prefix="*DEBUG: ";
    break;
  case MSG_INFO:
    prefix="**INFO: ";
    break;
  case MSG_WARNING:
    prefix="**WARNING: ";
    break;
  case MSG_ERROR:
    prefix="***ERROR: ";
    break;
  case MSG_CRITICAL_ERROR:
    prefix="****CRITICAL ERROR: ";
    break;
  default:
    cerr<<"I got an unknow error code: "<<msgCode_h<<" ("<<msg_h<<")"<<endl;
    exit(EXIT_FAILURE);
  }

  string message = prefix+msg_h;
  if (MTHREAD && MTHREAD->usingGUI()){
    MTHREAD->msgOut(msgCode_h, message);
  }
  else {
    string totalMsg = prefix+msg_h;
    cout<< totalMsg <<endl;
  }

  if(refreshGUI_h) {refreshGUI();}

  if(msgCode_h==MSG_CRITICAL_ERROR){
    if (MTHREAD && MTHREAD->usingGUI()){
      throw(2);
    }
    else {
      //throw(2);
      exit(EXIT_FAILURE);
    }
  }
}

void
BaseClass::refreshGUI()const{
  if (MTHREAD && MTHREAD->usingGUI()){
    MTHREAD->refreshGUI();
  }
}

int
BaseClass::s2i ( const string &string_h) const {
  if (string_h == "") return 0;
  int valueAsInteger;
  stringstream ss(string_h);
  ss >> valueAsInteger;
  return valueAsInteger;
  /*
  // I can't use stoi as of bug in MinGW
  try {
    return stoi(string_h);
  } catch (...) {
    if (string_h == "") return 0;
    else {
      msgOut(MSG_CRITICAL_ERROR,"Conversion string to integer failed. Some problems with the data? (got\""+string_h+"\")");
    }
  }
  return 0;
  */

}

double
BaseClass::s2d (const string& string_h) const {
  if (string_h == "") return 0.;
  double valueAsDouble;
  istringstream totalSString( string_h );
  totalSString >> valueAsDouble;
  return valueAsDouble;
  /*
  if (string_h == "") return 0.;
  try {
    return stod(string_h); // stod want dot as decimal separator in console mode and comma in gui mode. Further the decimal digits left are only 2 !!
  } catch (...) {
    if (string_h == "") return 0.;
    else {
      msgOut(MSG_CRITICAL_ERROR,"Conversion string to double failed. Some problems with the data? (got\""+string_h+"\")");
    }
  }
  return 0.;
  */
}


/// Includes comma to dot conversion if needed.
double
BaseClass::s2d (const  string& string_h, const bool& replaceComma) const {
  if(replaceComma){
    string valueAsString = string_h;
    // replace commas with dots. This is not needed when directly reading the input nodes as double, as the Qt function to Double does the same.
    replace(valueAsString.begin(), valueAsString.end(), ',', '.');
    return s2d(valueAsString);
  }
  return s2d(string_h);
  msgOut(MSG_CRITICAL_ERROR, "debug me please!");
  return 0.;
}

/// Includes conversion checks.
bool
BaseClass::s2b (const string& string_h) const {
  if (string_h == "true" || string_h == "vero" || string_h == "TRUE" || string_h == "1" || string_h == "True")
```

```
    return true;
  else if (string_h == "false" || string_h == "falso" || string_h == "FALSE" || string_h == "0" || string_h == "" || string_h == "False")
    return false;

  msgOut(MSG_CRITICAL_ERROR,"Conversion string to bool failed. Some problems with the data? (got\""+string_h+"\")");
  return true;
}

string
BaseClass::i2s (const int& int_h) const{
  //ostringstream out;
  //out<<int_h;
  //return out.str();
  char  outChar[24];
  snprintf ( outChar, sizeof(outChar), "%d", int_h );
  return string(outChar);
}

string
BaseClass::d2s (const double& double_h) const{
  //ostringstream out;
  //out<<double_h;
  //return out.str();
  char  outChar[24];
  snprintf ( outChar, sizeof(outChar), "%f", double_h );
  return string(outChar);
}

string
BaseClass::b2s (const bool& bool_h) const{
  if (bool_h) return "true";
  else return "false";
}

vector <int>
BaseClass::s2i(const  vector <string>& string_h) const{
  vector <int> valuesAsInteger;
  for (uint i=0;i<string_h.size();i++){
    valuesAsInteger.push_back(s2i(string_h[i]));
  }
  return valuesAsInteger;
}

/// Includes comma to dot conversion if needed.
vector <double>
BaseClass::s2d (const vector <string>& string_h, const bool& replaceComma) const{
  vector <double> valuesAsDouble;
  for (uint i=0;i<string_h.size();i++){
    if(replaceComma){
      string valueAsString = string_h[i];
      // replace commas with dots. This is not needed when directly reading the input nodes as double, as the Qt function to Double does the same.
      replace(valueAsString.begin(), valueAsString.end(), ',', '.');
      valuesAsDouble.push_back(s2d(valueAsString));
    } else {
      valuesAsDouble.push_back(s2d(string_h[i]));
    }
  }
  return valuesAsDouble;
}

/// Includes conversion checks.
vector <bool>
BaseClass::s2b(const vector <string> &string_h) const{
  vector <bool> valuesAsBool;
  for (uint i=0;i<string_h.size();i++){
    valuesAsBool.push_back(s2b(string_h[i]));
  }
  return valuesAsBool;
}

vector <string>
BaseClass::i2s (const vector <int> &int_h) const{
  vector <string> valuesAsString;
  for (uint i=0;i<int_h.size();i++){
        valuesAsString.push_back(i2s(int_h[i]));
  }
  return valuesAsString;
}

vector <string>
BaseClass::d2s (const vector <double> &double_h) const{
  vector <string> valuesAsString;
  for (uint i=0;i<double_h.size();i++){
        valuesAsString.push_back(d2s(double_h[i]));
  }
  return valuesAsString;
}

vector <string>
BaseClass::b2s (const vector <bool> &bool_h) const{
  vector <string> valuesAsString;
  for (uint i=0;i<bool_h.size();i++){
    if(bool_h[i]) valuesAsString.push_back("true");
    else valuesAsString.push_back("false");
  }
  return valuesAsString;
}


int
BaseClass::getType(const string &type_h) const{
  int toReturn=0;
  if (type_h == "int")         toReturn = TYPE_INT;
  else if (type_h == "double") toReturn = TYPE_DOUBLE;
  else if (type_h == "string") toReturn = TYPE_STRING;
  else if (type_h == "bool")   toReturn = TYPE_BOOL;
  else msgOut(MSG_CRITICAL_ERROR, "Unknow type "+type_h+".");
  return toReturn;
}


template<typename T> std::string
BaseClass::toString(const T& x) const {
  std::ostringstream oss;
  oss << x;
  return oss.str();
}


double
BaseClass::normSample (const double& avg, const double& stdev, const double& minval, const double& maxval) const{
  if(minval != NULL  && maxval != NULL){
    if (maxval <= minval){
      msgOut(MSG_CRITICAL_ERROR,"Error in normSample: the maxvalue is lower than the minvalue");
    }
  }
  for(;;){
    normal_distribution<double> d(avg,stdev);
    double c = d(*MTHREAD->gen);
    if( (minval == NULL || c >= minval) && (maxval == NULL || c <= maxval) ){
      return c;
    }
  }
  return minval;
}


template<typename T> T
BaseClass::stringTo(const std::string& s) const {
  std::istringstream iss(s);
  T x;
  iss >> x;
  return x;
}

int
BaseClass::vSum (const vector <vector<int> > &vector_h) const{
  int toReturn = 0;
  for(vector < vector<int> >::const_iterator j=vector_h.begin();j!=vector_h.end();++j){
    toReturn += accumulate(j->begin(),j->end(),0);
  }
  return toReturn;
}

double
BaseClass::vSum (const vector<vector<double> > &vector_h) const{
  double toReturn = 0.0;
  for(vector < vector<double> >::const_iterator j=vector_h.begin();j!=vector_h.end();++j){
    toReturn += accumulate(j->begin(),j->end(),0.0);
  }
  return toReturn;
}

void
BaseClass::tokenize(const string& str, vector<string>& tokens, const string& delimiter) const {
    // Skip delimiters at beginning.
    string::size_type lastPos = str.find_first_not_of(delimiter, 0);
    // Find first "non-delimiter".
    string::size_type pos     = str.find_first_of(delimiter, lastPos);

    while (string::npos != pos || string::npos != lastPos)
    {
        // Found a token, add it to the vector.
        tokens.push_back(str.substr(lastPos, pos - lastPos));
        // Skip delimiters.  Note the "not_of"
        lastPos = str.find_first_not_of(delimiter, pos);
        // Find next "non-delimiter"
        pos = str.find_first_of(delimiter, lastPos);
    }
}

void
BaseClass::untokenize(string &str, vector<string>& tokens, const string& delimiter) const {
  // add initial loken in str is not empty
  if(str != ""){
    str += delimiter;
  }
  for(int i=0;i<tokens.size();i++){
    str += tokens[i];
    // don't add final delimiter
    if(i != (tokens.size()-1)){
       str += delimiter;
    }
  }
}

////////////////////////////////////// OTHER CLASSES THAN BASECLASS //////////////////
/// iskey class ///
iskey::iskey(){
 i = 0;
 s = "";
}
iskey::iskey(int i_h, string s_h){
 i = i_h;
 s = s_h;
}

iskey::~iskey(){

}

bool
iskey::operator == (const iskey & op2) const{
  if(op2.i == i && op2.s == s){
    return true;
  }
  return false;
}

bool
iskey::operator != (const iskey & op2) const{
  if(op2.i == i && op2.s == s){
    return false;
  }
  return true;
}

bool
iskey::operator < (const iskey & op2) const{
  if (i < op2.i ) return true;
  if (i == op2.i) {
    if (s < op2.s) return true;
  }
  return false;
}

bool
iskey::operator > (const iskey & op2) const{
  if (i > op2.i ) return true;
  if (i == op2.i) {
    if (s > op2.s) return true;
  }
  return false;
}

bool
iskey::operator <= (const iskey & op2) const{
  if (i < op2.i ) return true;
  if (i == op2.i) {
    if (s <= op2.s) return true;
  }
  return false;
}

```

```
bool
iskey::operator >= (const iskey & op2) const{
  if (i > op2.i ) return true;
  if (i == op2.i) {
    if (s >= op2.s) return true;
  }
  return false;
}

/// iiskey class (note the double ii) ///
iiskey::iiskey(){
 i = 0;
 i2 = 0;
 s = "";
}
iiskey::iiskey(int i_h, int i2_h, string s_h){
 i  = i_h;
 i2 = i2_h;
 s  = s_h;
}

iiskey::~iiskey(){

}

bool
iiskey::operator == (const iiskey & op2) const{
  if(op2.i == i && op2.i2 == i2 && op2.s == s){
    return true;
  }
  return false;
}

bool
iiskey::operator != (const iiskey & op2) const{
  if(op2.i == i && op2.i2 == i2 && op2.s == s){
    return false;
  }
  return true;
}

bool
iiskey::operator < (const iiskey & op2) const{
  if (i < op2.i ) {return true;}
  if (i == op2.i) {
    if (i2 < op2.i2 ) {return true;}
    if (i2 == op2.i2){
      if (s < op2.s) {return true;}
    }
  }
  return false;
}

bool
iiskey::operator > (const iiskey & op2) const{
  if (i > op2.i ) {return true;}
  if (i == op2.i) {
    if (i2 > op2.i2 ) {return true;}
    if (i2 == op2.i2){
      if (s > op2.s) {return true;}
    }
  }
  return false;
}

bool
iiskey::operator <= (const iiskey & op2) const{
  if (i < op2.i ) {return true;}
  if (i == op2.i) {
    if (i2 < op2.i2 ) {return true;}
    if (i2 == op2.i2){
      if (s <= op2.s) {return true;}
    }
  }
  return false;
}

bool
iiskey::operator >= (const iiskey & op2) const{
  if (i > op2.i ) {return true;}
  if (i == op2.i) {
    if (i2 > op2.i2 ) {return true;}
    if (i2 == op2.i2){
      if (s >= op2.s) {return true;}
    }
  }
  return false;
```

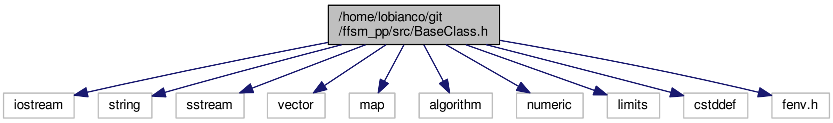
```
}

/// iisskey class (note the double ii and double ss) ///
iisskey::iisskey(){
 i = 0;
 i2 = 0;
 s = "";
 s2= "";
}
iisskey::iisskey(int i_h, int i2_h, string s_h, string s2_h){
 i  = i_h;
 i2 = i2_h;
 s  = s_h;
 s2 = s2_h;
}

iisskey::~iisskey(){

}

bool
iisskey::operator == (const iisskey & op2) const{
  if(op2.i == i && op2.i2 == i2 && op2.s == s && op2.s2 == s2){
    return true;
  }
  return false;
}

bool
iisskey::operator != (const iisskey & op2) const{
  if(op2.i == i && op2.i2 == i2 && op2.s == s && op2.s2 == s2){
    return false;
  }
  return true;
}

bool
iisskey::operator < (const iisskey & op2) const{
  if (i < op2.i ) {return true;}
  if (i == op2.i) {
    if (i2 < op2.i2 ) {return true;}
    if (i2 == op2.i2){
      if (s < op2.s) {return true;}
      if (s == op2.s){
        if (s2 < op2.s2) {return true;}
      }
    }
  }
  return false;
}

bool
iisskey::operator > (const iisskey & op2) const{
  if (i > op2.i ) {return true;}
  if (i == op2.i) {
    if (i2 > op2.i2 ) {return true;}
    if (i2 == op2.i2){
      if (s > op2.s) {return true;}
      if (s == op2.s){
        if (s2 > op2.s2) {return true;}
      }
    }
  }
  return false;
}

bool
iisskey::operator <= (const iisskey & op2) const{
  if (i < op2.i ) {return true;}
  if (i == op2.i) {
    if (i2 < op2.i2 ) {return true;}
    if (i2 == op2.i2){
      if (s < op2.s) {return true;}
      if (s == op2.s){
        if (s2 <= op2.s2) {return true;}
      }
    }
  }
  return false;
}

bool
iisskey::operator >= (const iisskey & op2) const{
  if (i > op2.i ) {return true;}
  if (i == op2.i) {
    if (i2 > op2.i2 ) {return true;}
    if (i2 == op2.i2){
      if (s > op2.s) {return true;}
      if (s == op2.s){
        if (s2 >= op2.s2) {return true;}
      }
    }
  }
  return false;
}

bool
iisskey::filter(const iisskey & key_h) const{
  if( (key_h.i == NULL  || key_h.i==i)     &&
      (key_h.i2 == NULL || key_h.i2==i2)  &&
      (key_h.s == ""  || key_h.s==s)     &&
      (key_h.s2 == "" || key_h.s2==s2)      ) return true;
  return false;
}

string
iisskey::print() const{
    char  outChar1[24];
    char  outChar2[24];
    snprintf ( outChar1, sizeof(outChar1), "%d", i);
    snprintf ( outChar2, sizeof(outChar2), "%d", i2);
    return string(outChar1)+'\t'+string(outChar2)+'\t'+s+'\t'+s2;

}
```

5.45 /home/lobianco/git/ffsm_pp/src/BaseClass.h File Reference

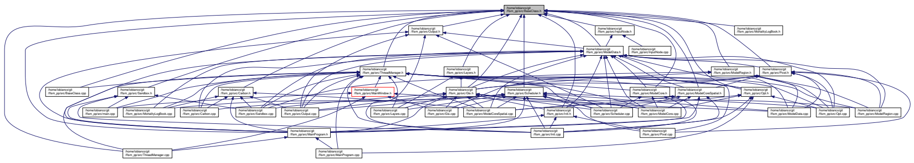
This file is the header of BaseClass and it is included by ALL compiled code.

```
#include <iostream>
#include <string>
#include <sstream>
#include <vector>
#include <map>
#include <algorithm>
#include <numeric>
#include <limits>
#include <cstddef>
#include <fenv.h>
```

Include dependency graph for BaseClass.h:



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



Classes

- class iskey

    Class to provide a simple integer-string key to be used in std maps.
- class iiskey

    Class to provide a simple integer-integer-string key in std maps.
- class iisskey

    Class to provide a simple integer-integer-string-string key in std maps.
- class BaseClass

    Base class for the regmas application.

Macros

- #define M_PI 3.14159265358979323846264338327950
- #define M_LN2 0.693147180559945309417232121458
- #define M_LN10 2.302585092994045684017991454684
- #define PROD_ALL "PROD_ALL"

    All primary and transformed products.
- #define PROD_PRI "PROD_PRI"

    Primary products.
- #define PROD_SEC "PROD_SEC"

    Secondary products.
- #define DIAM_ALL "DIAM_ALL"

    All diameter classes.
- #define DIAM_PROD "DIAM_PROD"

    Diameter classes used for production (e.g. excluded the first one)
- #define DIAM_FIRST "DIAM_FIRST_CLASS"

    First diameter class (NOT used for production)
- #define FT_ALL "FT_ALL"

    All forest types.
- #define LBOUND_MIN -20000000000000000000.0

    Lower bound in optimisation -10^ 19.
- #define UBOUND_MAX 20000000000000000000.0

    Upper bound in optimisation 10^ 19.

Enumerations

- enum messageType {
  MSG_NO_MSG = 0, MSG_DEBUG = 1, MSG_INFO = 2, MSG_WARNING = 3,
  MSG_ERROR = 4, MSG_CRITICAL_ERROR = 5 }

    Type of message to be printed.
- enum dataType { TYPE_INT =0, TYPE_DOUBLE =1, TYPE_STRING =2, TYPE_BOOL =3 }

    Type of data requested.
- enum dataRequest {
  DATA_NOW =-1, DATA_INIT = -2, DATA_ERROR = -99999999999, OP_SUM =1,
  OP_AVG =5 }

    A generic enum to deal with data requests.
- enum outputVerbosity {
  OUTVL_NONE =0, OUTVL_AGGREGATED =10, OUTVL_DETAILED =15, OUTVL_MAPS =18,
  OUTVL_BINMAPS =20, OUTVL_ALL =25 }

    Verbosity level of the output.

- enum domains {
DOM_PRI_PR =1, DOM_SEC_PR =2, DOM_ALL_PR =3, DOM_R2_PRI_PR =4,
DOM_R2_SEC_PR =5, DOM_R2_ALL_PR =6, DOM_SCALAR =7, DOM_PRI_PR_ALLCOMBS =8 }

  Domain associated to a variable or a constrain in the optimisation of the market module.
- enum carbonStocks { STOCK_INV =1, STOCK_EXTRA =2, STOCK_PRODUCTS =3 }

  Carbon stocks.
- enum emissionType { EM_ENSUB =4, EM_MATSUB =5, EM_FOROP =6 }

  Emission types.
- enum contrainDirection { CONSTR_EQ =1, CONSTR_LE0 =2, CONSTR_GE0 =3 }
- enum varType { VAR_VOL =1, VAR_AREA =2, VAR_IN =3 }
- enum boundType { LBOUND =1, UBOUND =2 }

5.45.1 Detailed Description

This file is the header of BaseClass and it is included by ALL compiled code.

It contains also global enum and macro definitions that can be used anywhere in the code. If the code require some "case" parameter, put the cases in the enum here. DON'T USE NEGATIVE NUMBERS in the enums, as often negative numbers have a different meaning !

Definition in file BaseClass.h.

5.45.2 Macro Definition Documentation

5.45.2.1 #define DIAM_ALL "DIAM_ALL"

All diameter classes.

Definition at line 154 of file BaseClass.h.

Referenced by ModelData::getForData(), Carbon::getStock(), ModelCoreSpatial::initialiseCarbonModule(), Output::printForestData(), ModelCoreSpatial::registerCarbonEvents(), ModelCore::runManagementModule(), ModelData::setForData(), and ModelCore::updateMapAreas().

5.45.2.2 #define DIAM_FIRST "DIAM_FIRST_CLASS"

First diameter class (NOT used for production)

Definition at line 160 of file BaseClass.h.

Referenced by ModelData::getForData(), and ModelData::setForData().

5.45.2.3 #define DIAM_PROD "DIAM_PROD"

Diameter classes used for production (e.g. excluded the first one)

Definition at line 157 of file BaseClass.h.

Referenced by ModelData::getForData(), and ModelData::setForData().

5.45.2.4 #define FT_ALL "FT_ALL"

All forest types.

Definition at line 163 of file BaseClass.h.

Referenced by ModelData::getForData(), ModelCore::runManagementModule(), and ModelData::setForData().

5.45.2.5 #define LBOUND_MIN -20000000000000000000.0

Lower bound in optimisation -10^19.

Definition at line 168 of file BaseClass.h.

5.45.2.6 #define M_LN10 2.3025850929940456840179914546844

Definition at line 140 of file BaseClass.h.

5.45.2.7 #define M_LN2 0.69314718055994530941723212145818

Definition at line 136 of file BaseClass.h.

5.45.2.8 #define M_PI 3.14159265358979323846264338327950

Definition at line 132 of file BaseClass.h.

5.45.2.9 #define PROD_ALL "PROD_ALL"

All primary and transformed products.

Definition at line 145 of file BaseClass.h.

Referenced by ModelData::getProdData(), and ModelData::setProdData().

5.45.2.10 #define PROD_PRI "PROD_PRI"

Primary products.

Definition at line 148 of file BaseClass.h.

Referenced by ModelData::getProdData(), and ModelData::setProdData().

5.45.2.11 #define PROD_SEC "PROD_SEC"

Secondary products.

Definition at line 151 of file BaseClass.h.

Referenced by ModelData::getProdData(), and ModelData::setProdData().

5.45.2.12 #define UBOUND_MAX 20000000000000000000.0

Upper bound in optimisation 10^19.

Definition at line 171 of file BaseClass.h.

5.45.3 Enumeration Type Documentation

5.45.3.1 enum boundType

Enumerator

LBOUND

UBOUND

Definition at line 124 of file BaseClass.h.

```
00124                 {
00125   LBOUND                  =1,
00126   UBOUND                  =2
00127 };
```

5.45.3.2 enum carbonStocks

Carbon stocks.

Enumerator

STOCK_INV Invetoried biomass (live and death tree logs)

STOCK_EXTRA Extra biomass (soils, branches..)

STOCK_PRODUCTS Biomass in forest products (sawns, pannels..)

Definition at line 100 of file BaseClass.h.

```
00100                   {
00101   STOCK_INV               =1,             ///< Invetoried biomass (live and death tree logs)
00102   STOCK_EXTRA             =2,             ///< Extra biomass (soils, branches..)
00103   STOCK_PRODUCTS          =3,             ///< Biomass in forest products (sawns, pannels..)
00104 };
```

5.45.3.3 enum contrainDirection

Enumerator

CONSTR_EQ

CONSTR_LE0

CONSTR_GE0

Definition at line 112 of file BaseClass.h.

```
00112                       {
00113   CONSTR_EQ               =1, // constrain of type equality
00114   CONSTR_LE0              =2, // constrain of type lower or equal than 0
00115   CONSTR_GE0              =3, // constrain of type greater or equal 0
00116 };
```

5.45.3.4 enum dataRequest

A generic enum to deal with data requests.

Enumerator

DATA_NOW The required data is for the current year.

DATA_INIT Setting a data request for the init period.

DATA_ERROR There is an error in retrieving the data.

OP_SUM Perform a SUM operation.

OP_AVG Perform an AVERAGE operation.

Definition at line 71 of file BaseClass.h.

```
00071                  {
00072   DATA_NOW               =-1,            ///< The required data is for the current year
00073   DATA_INIT              = -2,           ///< Setting a data request for the init period
00074   DATA_ERROR             = -99999999999, ///< There is an error in retrieving the data
00075   // operations possible in certain contexts
00076   OP_SUM                 =1,             ///< Perform a SUM operation
00077   OP_AVG                 =5,             ///< Perform an AVERAGE operation
00078 };
```

5.45.3.5 enum dataType

Type of data requested.

Enumerator

TYPE_INT The required data is an integer.

TYPE_DOUBLE The required data is a double.

TYPE_STRING The required data is a string.

TYPE_BOOL The required data is a bool.

Definition at line 64 of file BaseClass.h.

```
00064              {
00065   TYPE_INT               =0,             ///< The required data is an integer
00066   TYPE_DOUBLE            =1,             ///< The required data is a double
00067   TYPE_STRING            =2,             ///< The required data is a string
00068   TYPE_BOOL              =3,             ///< The required data is a bool
00069 };
```

5.45.3.6 enum domains

Domain associated to a variable or a constrain in the optimisation of the market module.

Enumerator

DOM_PRI_PR Primary products // domain of variables and constrains: primary, secondary, all products or all products over r2 couple regions (in-country commercial flows)

DOM_SEC_PR Secondary products.

DOM_ALL_PR All products (primary+secondary)

DOM_R2_PRI_PR Primary products over r2 couple regions (in-country commercial flows)

DOM_R2_SEC_PR Secondary products over r2 couple regions (in-country commercial flows)

DOM_R2_ALL_PR All products over r2 couple regions (in-country commercial flows)

DOM_SCALAR Scalar variable (not used)

DOM_PRI_PR_ALLCOMBS All possible combinations of primary products (2^ number of primary products)

Definition at line 89 of file BaseClass.h.

```
00089               {
00090   DOM_PRI_PR             =1,             ///< Primary products // domain of variables and
       constrains: primary, secondary, all products or all products over r2 couple regions (in-country commercial flows)
00091   DOM_SEC_PR             =2,             ///< Secondary products
00092   DOM_ALL_PR             =3,             ///< All products (primary+secondary)
00093   DOM_R2_PRI_PR          =4,             ///< Primary products over r2 couple regions
       (in-country commercial flows)
00094   DOM_R2_SEC_PR          =5,             ///< Secondary products over r2 couple regions
       (in-country commercial flows)
00095   DOM_R2_ALL_PR          =6,             ///< All products over r2 couple regions (in-country
       commercial flows)
00096   DOM_SCALAR             =7,             ///< Scalar variable (not used)
00097   DOM_PRI_PR_ALLCOMBS    =8,             ///< All possible combinations of primary
       products (2^ number of primary products)
00098 };
```

5.45.3.7 enum emissionType

Emission types.

Enumerator

EM_ENSUB Energy substitution.

EM_MATSUB Material substitution.

EM_FOROP Flow from forest operations.

Definition at line 106 of file BaseClass.h.

```
00106                    {
00107   EM_ENSUB               =4,             ///< Energy substitution
00108   EM_MATSUB              =5,             ///< Material substitution
00109   EM_FOROP               =6,             ///< Flow from forest operations
00110 };
```

5.45.3.8 enum messageType

Type of message to be printed.

Enumerator

MSG_NO_MSG Do not actually output any message.

MSG_DEBUG Print a debug message, normally filtered out.

MSG_INFO Print an INFO message.

MSG_WARNING Print a WARNING message.

MSG_ERROR Print an ERROR message, but don't stop the model.

MSG_CRITICAL_ERROR Print an error message and stop the model.

Definition at line 54 of file BaseClass.h.

```
00054                 {
00055
00056   MSG_NO_MSG            = 0,            ///< Do not actually output any message
00057   MSG_DEBUG             = 1,            ///< Print a debug message, normally filtered out
00058   MSG_INFO              = 2,            ///< Print an INFO message
00059   MSG_WARNING           = 3,            ///< Print a WARNING message
00060   MSG_ERROR             = 4,            ///< Print an ERROR message, but don't stop the model
00061   MSG_CRITICAL_ERROR    = 5,            ///< Print an error message and stop the model
00062 };
```

5.45.3.9 enum outputVerbosity

Verbosity level of the output.

Enumerator

OUTVL_NONE Output verbosity level none.

OUTVL_AGGREGATED Output verbosity level print aggregated output (e.g. optimisation log)

OUTVL_DETAILED Output verbosity level print (also) detailed output.

OUTVL_MAPS Output verbosity level print (also) the maps in ascii grid format.

OUTVL_BINMAPS Output verbosity level print (also) binary (png) maps.

OUTVL_ALL Output verbosity level print everything.

Definition at line 80 of file BaseClass.h.

```
00080                      {
00081   OUTVL_NONE             =0,            ///< Output verbosity level none
00082   OUTVL_AGGREGATED       =10,           ///< Output verbosity level print aggregated
 output (e.g. optimisation log)
00083   OUTVL_DETAILED         =15,           ///< Output verbosity level print (also) detailed
 output
00084   OUTVL_MAPS             =18,           ///< Output verbosity level print (also) the maps in
 ascii grid format
00085   OUTVL_BINMAPS          =20,           ///< Output verbosity level print (also) binary (png)
 maps
00086   OUTVL_ALL              =25,           ///< Output verbosity level print everything
00087 };
```

5.45.3.10 enum varType

Enumerator

VAR_VOL

VAR_AREA

VAR_IN

Definition at line 118 of file BaseClass.h.

```
00118              {
00119   VAR_VOL                =1,
00120   VAR_AREA               =2,
00121   VAR_IN                 =3
00122 };
```

5.46 BaseClass.h

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/

/**
 * \file BaseClass.h
 * \brief This file is the header of BaseClass and it is included by ALL compiled code.
 *
 * It contains also global enum and macro definitions that can be used anywhere in the code.
 * If the code require some "case" parameter, put the cases in the enum here.
 * DON’T USE NEGATIVE NUMBERS in the enums, as often negative numbers have a different meaning !
 *
 */

#ifndef BASECLASSBASECLASS_H
#define BASECLASSBASECLASS_H

#include <iostream>
#include <string>
#include <sstream>
#include <vector>
#include <map>
#include <algorithm>
#include <numeric>
#include <limits>
#include <cstddef>
#include <fenv.h> // for division by zero runtime error

// regmas headers...

class ThreadManager;

using namespace std;

/// Type of message to be printed
enum messageType{

  MSG_NO_MSG            = 0,             ///< Do not actually output any message
  MSG_DEBUG             = 1,             ///< Print a debug message, normally filtered out
  MSG_INFO              = 2,             ///< Print an INFO message
  MSG_WARNING           = 3,             ///< Print a WARNING message
  MSG_ERROR             = 4,             ///< Print an ERROR message, but don't stop the model
  MSG_CRITICAL_ERROR    = 5,             ///< Print an error message and stop the model
};
/// Type of data requested
enum dataType{
  TYPE_INT               =0,             ///< The required data is an integer
  TYPE_DOUBLE            =1,             ///< The required data is a double
  TYPE_STRING            =2,             ///< The required data is a string
  TYPE_BOOL              =3,             ///< The required data is a bool
};
/// A generic enum to deal with data requests
enum dataRequest {
  DATA_NOW               =-1,            ///< The required data is for the current year
  DATA_INIT             = -2,            ///< Setting a data request for the init period
  DATA_ERROR            = -99999999999, ///< There is an error in retrieving the data
  // operations possible in certain contexts
  OP_SUM                 =1,             ///< Perform a SUM operation
  OP_AVG                 =5,             ///< Perform an AVERAGE operation
};
/// Verbosity level of the output
enum outputVerbosity {
  OUTVL_NONE             =0,             ///< Output verbosity level none
  OUTVL_AGGREGATED       =10,            ///< Output verbosity level print aggregated
 output (e.g. optimisation log)
  OUTVL_DETAILED         =15,            ///< Output verbosity level print (also) detailed
 output
  OUTVL_MAPS             =18,            ///< Output verbosity level print (also) the maps in
 ascii grid format
  OUTVL_BINMAPS          =20,            ///< Output verbosity level print (also) binary (png)
 maps
  OUTVL_ALL              =25,            ///< Output verbosity level print everything
};
/// Domain associated to a variable or a constrain in the optimisation of the market module
enum domains {
  DOM_PRI_PR             =1,             ///< Primary products // domain of variables and
 constrains: primary, secondary, all products or all products over r2 couple regions (in-country commercial flows)
  DOM_SEC_PR             =2,             ///< Secondary products
  DOM_ALL_PR             =3,             ///< All products (primary+secondary)
  DOM_R2_PRI_PR          =4,             ///< Primary products over r2 couple regions
 (in-country commercial flows)
  DOM_R2_SEC_PR          =5,             ///< Secondary products over r2 couple regions
 (in-country commercial flows)
  DOM_R2_ALL_PR          =6,             ///< All products over r2 couple regions (in-country
 commercial flows)
  DOM_SCALAR             =7,             ///< Scalar variable (not used)
  DOM_PRI_PR_ALLCOMBS    =8,             ///< All possible combinations of primary
 products (2^ number of primary products)
};
/// Carbon stocks
enum carbonStocks {
  STOCK_INV              =1,             ///< Invetoried biomass (live and death tree logs)
  STOCK_EXTRA            =2,             ///< Extra biomass (soils, branches..)
  STOCK_PRODUCTS         =3,             ///< Biomass in forest products (sawns, pannels..)
};
/// Emission types
enum emissionType {
  EM_ENSUB               =4,             ///< Energy substitution
  EM_MATSUB              =5,             ///< Material substitution
  EM_FOROP               =6,             ///< Flow from forest operations
};

enum contrainDirection {
  CONSTR_EQ              =1, // constrain of type equality
  CONSTR_LE0             =2, // constrain of type lower or equal than 0
  CONSTR_GE0             =3, // constrain of type greater or equal 0
};

enum varType {
  VAR_VOL                =1,
  VAR_AREA               =2,
  VAR_IN                 =3
};

enum boundType {
  LBOUND                 =1,
  UBOUND                 =2
};

// mathematical defines (not correctly implemented in some compilers, namely MS VisualStudio..)

#ifndef M_PI
#define M_PI 3.14159265358979323846264338327950288
#endif

#ifndef M_LN2
#define M_LN2 0.693147180559945309417232121458
```

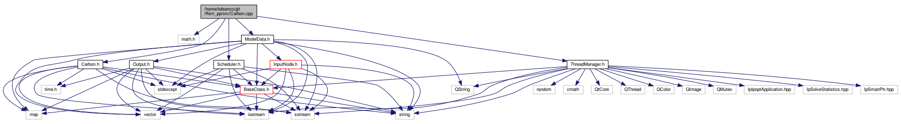
```
#endif

#ifndef M_LN10
#define M_LN10 2.30258509299404568401799145468
#endif

// some macro for specific keywords in the model
#ifndef PROD_ALL
#define PROD_ALL "PROD_ALL"           ///< All primary and transformed products
#endif
#ifndef PROD_PRI
#define PROD_PRI "PROD_PRI"           ///< Primary products
#endif
#ifndef PROD_SEC
#define PROD_SEC "PROD_SEC"           ///< Secondary products
#endif
#ifndef DIAM_ALL
#define DIAM_ALL "DIAM_ALL"           ///< All diameter classes
#endif
#ifndef DIAM_PROD
#define DIAM_PROD "DIAM_PROD"         ///< Diameter classes used for production (e.g. excluded the first one)
#endif
#ifndef DIAM_FIRST
#define DIAM_FIRST "DIAM_FIRST_CLASS" ///< First diameter class (NOT used for production)
#endif
#ifndef FT_ALL
#define FT_ALL "FT_ALL"               ///< All forest types
#endif

// Bounds for optimisation
#ifndef LBOUND_MIN
#define LBOUND_MIN -20000000000000000000.0 ///< Lower bound in optimisation -10^19
#endif
#ifndef UBOUND_MAX
#define UBOUND_MAX 20000000000000000000.0  ///< Upper bound in optimisation 10^19
#endif


/// Class to provide a simple integer-string key to be used in std maps
class iskey {
public:
                      iskey();
                      iskey(int i_h, string s_h);
                     ~iskey();
  bool                operator == (const iskey &op2) const;
  bool                operator != (const iskey &op2) const;
  bool                operator <  (const iskey &op2) const;
  bool                operator >  (const iskey &op2) const;
  bool                operator <= (const iskey &op2) const;
  bool                operator >= (const iskey &op2) const;
  int                                        i;
  string                                     s;
};

/// Class to provide a simple integer-integer-string key in std maps
class iiskey {
public:
                      iiskey();
                      iiskey(int i_h, int i2_h, string s_h);
                     ~iiskey();
  bool                operator == (const iiskey &op2) const;
  bool                operator != (const iiskey &op2) const;
  bool                operator <  (const iiskey &op2) const;
  bool                operator >  (const iiskey &op2) const;
  bool                operator <= (const iiskey &op2) const;
  bool                operator >= (const iiskey &op2) const;
  int                                        i;
  int                                        i2;
  string                                     s;

};

/// Class to provide a simple integer-integer-string-string key in std maps
class iisskey {
public:
                      iisskey();
                      iisskey(int i_h, int i2_h, string s_h, string s2_h);
                     ~iisskey();
  bool                filter(const iisskey & key_h)     const;
  string              print()                           const;
  bool                operator == (const iisskey &op2) const;
  bool                operator != (const iisskey &op2) const;
  bool                operator <  (const iisskey &op2) const;
  bool                operator >  (const iisskey &op2) const;
  bool                operator <= (const iisskey &op2) const;
  bool                operator >= (const iisskey &op2) const;
  int                                           i;
  int                                           i2;
  string                                        s;
  string                                        s2;
};

/// Base class for the regmas application.
/**
This class is the base class for all classes in regmas. \\
It provides common methods in all parts of the application for printing the output, converting strings vs. values or regularly "ping" the GUI. \\
@author Antonello Lobianco
*/

class BaseClass{

public:
                      BaseClass();
                     ~BaseClass();

  void                msgOut(const int& msgCode_h, const string& msg_h, const bool& refreshGUI_h=true) const; ///< Overloaded function to print the output log
  void                msgOut(const int& msgCode_h, const int& msg_h, const bool& refreshGUI_h=true) const; ///< Overloaded function to print the output log
  void                msgOut(const int& msgCode_h, const double& msg_h, const bool& refreshGUI_h=true) const; ///< Overloaded function to print the output log

  int                 s2i (const string& string_h)          const; ///<  string  to integer conversion
  double              s2d (const string& string_h) const; ///<  string  to double  conversion
  double              s2d (const string& string_h, const bool& replaceComma) const; ///<  string  to double conversion
  bool                s2b (const string& string_h)           const; ///<  string  to bool    conversion
  // as of 20120909 c++11 to_string(), stoi and stod functions not working in MinGw, as bug http://gcc.gnu.org/bugzilla/show_bug.cgi?id=52015
  // reverting to old code :-(
  //string              i2s (const int& int_h)       const {return to_string(int_h);};    ///<  integer to string  conversion
  //string              d2s (const double& double_h) const {return to_string(double_h);}; ///<  double  to string  conversion
  string              i2s (const int& int_h)                const; ///<  integer to string  conversion
  string              d2s (const double& double_h)          const; ///<  double  to string  conversion
  string              b2s (const bool& bool_h)              const; ///<  bool    to string  conversion

  vector<int>         s2i (const vector<string>& string_h) const; ///<  string  to integer conversion (vector)
  vector<double>      s2d (const vector<string>& string_h, const bool& replaceComma = false) const; ///< string to double conversion (vector)
  vector<bool>        s2b (const vector<string>& string_h) const; ///<  string  to bool    conversion (vector)
  vector<string>      i2s (const vector<int>& int_h)       const; ///<  integer to string  conversion (vector)
  vector<string>      d2s (const vector<double>& double_h) const; ///<  double  to string  conversion (vector)
  vector<string>      b2s (const vector<bool>& bool_h)     const; ///<  bool    to string  conversion (vector)

  int                 getType(const string &type_h) const; ///< Return a type according to enum TYPE_* from a string (eg: "string" -> TYPE_STRING (2))
  void                refreshGUI() const;                  ///< Ping to periodically return the control to the GUI

  //string intToString(int x);
  template<typename T> string toString(const T& x) const;      // string s = toString(x);
  template<typename T> T stringTo(const std::string& s) const; // e.g. int x = stringTo<int>("123");

  // vector and vector of vector sums
  int                   vSum(const vector<int>& vector_h) const {return accumulate(vector_h.begin(), vector_h.end(),0);};
  double                vSum(const vector<double>& vector_h) const {return accumulate(vector_h.begin(), vector_h.end(),0.);};
  int                   vSum(const vector< vector <int> >& vector_h) const;
  double                vSum(const vector< vector <double> >& vector_h) const;

  /// Tokenize a string using a delimiter (default is space)
  void                tokenize(const string& str, vector<string>& tokens, const string& delimiter = " ") const; // See also http://stackoverflow.com/questions/236129/split-a-string-in-c that could be faster
  void                untokenize(string &str, vector<string>& tokens, const string& delimiter = " ") const;

  /// Lookup a map for a value. Return the value starting from the key
  template <typename K, typename V> V findMap(const map <K, V> &mymap, const K& key, const int& error_level=MSG_CRITICAL_ERROR, const V &notFoundValue=numeric_limits<V>::min()) const{
    typename map<K, V>::const_iterator p;
    p=mymap.find(key);
    if(p != mymap.end()) {
      return p->second;
    }
    else {
```

```
    msgOut(error_level, "Error in finding a value in a map (no value found)");
    return notFoundValue;
  }
}

/// Change the value stored in a map given the key and the new value
template <typename K, typename V> void changeMapValue(map <K, V> &mymap, const K& key, const V& value, const int& error_level=MSG_CRITICAL_ERROR){
    typename map<K, V>::iterator p;
    p=mymap.find(key);
    if(p != mymap.end()) {
        p->second = value;
        return;
    }
    else {
        msgOut(error_level, "Error in finding a value in a map (no value found)");
    }
}

/// Increments a value stored in a map of the specified value, given the key
template <typename K, typename V> void incrMapValue(map <K, V> &mymap, const K& key, const V& value, const int& error_level=MSG_CRITICAL_ERROR){
    typename map<K, V>::iterator p;
    p=mymap.find(key);
    if(p != mymap.end()) {
        p->second = p->second + value;
        return;
    }
    else {
        msgOut(error_level, "Error in finding a value in a map (no value found)");
    }
}

/// Increments a value stored in a map of the specified value, given the key
template <typename K, typename V> void incrOrAddMapValue(map <K, V> &mymap, const K& key, const V& value){
    typename map<K, V>::iterator p;
    p=mymap.find(key);
    if(p != mymap.end()) {
        // We found the key, we gonna add the value..
        p->second = p->second + value;
        return;
    }
    else {
        // We didn't find the key, we gonna add it together with the value
        pair<K,V> myPair(key,value);
        mymap.insert(myPair);
    }
}

/// Reset all values stored in a map to the specified one
template <typename K, typename V> void resetMapValues(map <K, V> &mymap, const V& value){
    typename map<K, V>::iterator p;
    for(p=mymap.begin(); p!=mymap.end(); p++) {
        p->second =value;
    }
}

/// Returns a map built using the given vector and the given (scalar) value as keys/values pairs
template <typename K, typename V> map <K, V> vectorToMap(const vector <K>& keys, const V& value=0.0){
    map<K,V> returnMap;
    for(unsigned int i=0; i<keys.size();i++){
        pair<K,V> apair(keys[i],value);
        returnMap.insert(apair);
    }
    return returnMap;
}

/// Return a vector of content from a vector and a vector of positions (int)
template <typename T> vector <T> positionsToContent(const vector <T> & vector_h, const vector<int> &positions){
    vector <T> toReturn;
    for(uint i=0; i<positions.size(); i++){
        toReturn.push_back(vector_h.at(positions[i]));
    }
    return toReturn;
}

/// Debug a map
template <typename V> void debugMap(const map <iisskey, V> &mymap){
  iisskey mykey(NULL,NULL,"","");
  debugMap(mymap, mykey);
}
template <typename K, typename V> void debugMap(const map <K, V> &mymap, const K& key){
  cout<<"Debugging a map" << endl;
  for (auto const &themap: mymap) {
    if(themap.first.filter(key)){
     cout << themap.first.print() <<  '\t' << themap.second << endl;
    }
  }
}


/// Returns the position of the maximum element in the vector (the last one in case of multiple equivalent maxima)
template <typename K>  int getMaxPos (const vector <K> & v){
  return (minmax_element(v.begin(), v.end()).second - v.begin());
}
/// Returns the position of the minimum element in the vector (the first one in case of multiple equivalent minima)
template <typename K>  int getMinPos (const vector <K> & v){
  return (minmax_element(v.begin(), v.end()).first - v.begin());
}
/// Returns the value of the maximum element in the vector (the last one in case of multiple equivalent maxima)
template <typename K>  K getMax(const vector <K> & v){
  return *minmax_element(v.begin(), v.end()).second;
}
/// Returns the value of the minimum element in the vector (the first one in case of multiple equivalent minima)
template <typename K>  K getMin (const vector <K> & v){
  return *minmax_element(v.begin(), v.end()).first;
}
/// Returns the average of the elements in the vector
template <typename K>  double getAvg (const vector <K> & v){
  return v.size()==0 ? 0.0 : vSum(v)/ ( (double) v.size() );
}

/** Returns the sd of the elements of a vector. Default to sample sd.
*
* See http://stackoverflow.com/questions/7616511/calculate-mean-and-standard-deviation-from-a-vector-of-samples-in-c-using-boos
* where there is also an example for covariance
*/
template <typename K>  double getSd (const vector <K> & v, bool sample=true){
  if (v.size()==0) return 0.0;
  int sampleCorrection = sample==true?1:0;
  double sum = std::accumulate(std::begin(v), std::end(v), 0.0);
  double m =  sum / v.size();
  double accum = 0.0;
  std::for_each (std::begin(v), std::end(v), [&](const double d) {
    accum += (d - m) * (d - m);
  });
  double stdev = sqrt(accum / ( (double) (v.size()-sampleCorrection)));
  return stdev;
}

template <typename K>  int getPos (const K & element, const vector <K> & v, const int& msgCode_h=MSG_CRITICAL_ERROR){
  for(unsigned int i=0; i<v.size(); i++){
    if(v[i]== element) return i;
  }
  msgOut(msgCode_h, "Element not found in vector in getPos()");
  return -1;
}

template <typename K> bool inVector (const K & element, const vector <K> & v){
   for(unsigned int i=0; i<v.size(); i++){
     if(v[i]== element) return true;
   }
   return false;
}

/// Sample from a normal distribution with bounds. Slower (double time, but still you see the diff only after milion of loops).

/// It doesn't require the normal_distribution to be passed to it, but due to including MTHREAD its definition can't be placed
/// in the header and hence it can not be templated, so it works only with doubles.
double normSample (const double& avg, const double& stdev, const double& minval=NULL, const double& maxval=NULL) const;

/// Sample from a normal distribution with bounds. Faster (half time) as the normal_distribution is made only once.
template <typename K> K normSample (normal_distribution<K>& d, std::mt19937& gen, const K& minval=NULL, const K& maxval=NULL) const {
  if(minval != NULL  && maxval != NULL){
    if (maxval <= minval){
      msgOut(MSG_CRITICAL_ERROR,"Error in normSample: the maxvalue is lower than the minvalue");
    }
  }
  for(;;){
    K c = d(gen);
    if( (minval == NULL || c >= minval) && (maxval == NULL || c <= maxval) ){
      return c;
    }
  }
  return minval;
}




protected:

/**
* Through this pointer each derived subclass (the vast maiority of those used on FFSM) can "ask"
* for sending signals to the GUI, like append the log or modify the map.
*/
ThreadManager*                        MTHREAD; ///< Pointer to the Thread manager.
// ATTENTION
// I can't create member variables to host return values as these would break all the const mechanism..

private:
void msgOut2(const int& msgCode_h, const string& msg_h, const bool& refreshGUI_h) const; ///< Do the job of the overloaded functions

};




#endif
```

5.47 /home/lobianco/git/ffsm_pp/src/Carbon.cpp File Reference

```
#include <math.h>
#include "Carbon.h"
#include "ThreadManager.h"
#include "ModelData.h"
#include "Scheduler.h"
```

Include dependency graph for Carbon.cpp:



5.48 Carbon.cpp

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d'Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/

```

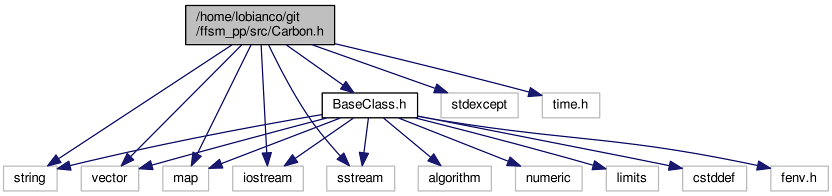
```
#include <math.h>       /* log */

#include "Carbon.h"
#include "ThreadManager.h"
#include "ModelData.h"
#include "Scheduler.h"



Carbon::Carbon(ThreadManager* MTHREAD_h){
  MTHREAD=MTHREAD_h;
}

Carbon::~Carbon(){
}


// ################# GET FUNCTIONS ################
/**
 * @param reg
 * @param stock_type
 * @return the Carbon stocked in a given sink
 *
 * For product sink:
 * - for primary products it includes the primary products exported out of the country, but not those exported to other regions or used in the region as
 *   these are assumed to be totally transformed to secondary products;
 * - for secondary products it includes those produced in the region from locally or regionally imported primary product plus those secondary products
 *   imported from other regions, less those exported to other regions. It doesn't include the secondary products imported from abroad the country.
 */
double
Carbon::getStock(const int & regId, const int & stock_type) const{
  double toReturn = 0.0;
  int currentYear = MTHREAD->SCD->getYear();
  int initialYear = MTHREAD->MD->getIntSetting("initialYear");
  switch (stock_type){
    case STOCK_PRODUCTS: {
      vector <string> priProducts = MTHREAD->MD->getStringVectorSetting("priProducts");
      vector <string> secProducts = MTHREAD->MD->getStringVectorSetting("secProducts");
      vector <string> allProducts = priProducts;
      allProducts.insert( allProducts.end(), secProducts.begin(), secProducts.end() );
      for(uint i=0;i<allProducts.size();i++){
        double coeff = MTHREAD->MD->getProdData("co2content_products",regId,allProducts[i],DATA_NOW,""); //  [kg CO2/m^3 wood]
        double life  = MTHREAD->MD->getProdData("avgLife_products",regId,allProducts[i],DATA_NOW,"");
        //for(int y=currentYear;y>currentYear-life;y--){ // ok
        //  iiskey key(y,regId,allProducts[i]);
        //  toReturn += findMap(products,key,MSG_NO_MSG,0.0)*coeff/1000;
        //}
        for(int y=(initialYear-100);y<=currentYear;y++){
          iiskey key(y,regId,allProducts[i]);
          double originalStock = findMap(products,key,MSG_NO_MSG,0.0);
          double remainingStock = getRemainingStock(originalStock,life,currentYear-y);
          toReturn += remainingStock*coeff/1000;
        }
      }
      break;
    }
    case STOCK_INV:{
      vector <string> fTypes = MTHREAD->MD->getForTypeIds();
      for(uint i=0;i<fTypes.size();i++){
        // units:
        // co2content_inventory: [Kg CO2 / m^3 wood]
        // co2content_extra:     [Kg CO2 / m^3 inventaried wood]
        double coeff = MTHREAD->MD->getForData("co2content_inventory",regId,fTypes[i],"",DATA_NOW); //  [kg CO2/m^3 wood]
        double life  = MTHREAD->MD->getForData("avgLive_deathBiomass_inventory",regId,fTypes[i],"",DATA_NOW);
        // PART A: from death biomass..
        //for(int y=currentYear;y>currentYear-life;y--){ // ok
        //  iiskey key(y,regId,fTypes[i]);
        //  toReturn += findMap(deathBiomassInventory,key,MSG_NO_MSG)*coeff/1000;
        //}
        for(int y=(initialYear-100);y<=currentYear;y++){
          iiskey key(y,regId,fTypes[i]);
          double originalStock = findMap(deathBiomassInventory,key,MSG_NO_MSG,0.0);
          double remainingStock = getRemainingStock(originalStock,life,currentYear-y);
          toReturn += remainingStock*coeff/1000;
        }

        // PART B: from inventory volumes
        toReturn += MTHREAD->MD->getForData("vol",regId,fTypes[i],DIAM_ALL,DATA_NOW)*coeff/1000;
      }
      break;

    }
    case STOCK_EXTRA:{
      vector <string> fTypes = MTHREAD->MD->getForTypeIds();
      for(uint i=0;i<fTypes.size();i++){
        // units:
        // co2content_inventory: [Kg CO2 / m^3 wood]
        // co2content_extra:     [Kg CO2 / m^3 inventaried wood]
        double coeff = MTHREAD->MD->getForData("co2content_extra",regId,fTypes[i],"",DATA_NOW); //  [kg CO2/m^3 wood]
        double life  = MTHREAD->MD->getForData("avgLive_deathBiomass_extra",regId,fTypes[i],"",DATA_NOW);
        // PART A: from death biomass..
        //for(int y=currentYear;y>currentYear-life;y--){ // ok
        //  iiskey key(y,regId,fTypes[i]);
        //  toReturn += findMap(deathBiomassExtra,key,MSG_NO_MSG),0.0*coeff/1000;
        //}
        for(int y=(initialYear-100);y<=currentYear;y++){
          iiskey key(y,regId,fTypes[i]);
          double originalStock = findMap(deathBiomassExtra,key,MSG_NO_MSG,0.0);
          double remainingStock = getRemainingStock(originalStock,life,currentYear-y);
          toReturn += remainingStock*coeff/1000;
        }
        // PART B: from inventory volumes
        double extraBiomass_ratio = MTHREAD->MD->getForData("extraBiomass_ratio",regId,fTypes[i],"",DATA_NOW);
        toReturn += MTHREAD->MD->getForData("vol",regId,fTypes[i],DIAM_ALL,DATA_NOW)*extraBiomass_ratio*coeff/1000;
      }
      break;
    }
    default:
      msgOut(MSG_CRITICAL_ERROR,"Unexpected stock_type in function getStock");
  }
  return toReturn;
}


double
Carbon::getCumSavedEmissions(const int & regId, const int & em_type) const{
  switch (em_type){
    case EM_ENSUB:
      return findMap(cumSubstitutedEnergy, regId);
      break;
    case EM_MATSUB:
      return findMap(cumSubstitutedMaterial, regId);
      break;
    case EM_FOROP:
      return -findMap(cumEmittedForOper, regId);
      break;
    default:
      msgOut(MSG_CRITICAL_ERROR,"Unexpected em_type in function getCumSavedEmissions");
  }
  return 0.0;
}

// ################ INITIALISE FUNCTIONS ###############

void
Carbon::initialiseEmissionCounters(){
  vector<int> regIds = MTHREAD->MD->getRegionIds(2);
  for (uint i=0;i<regIds.size();i++){
    pair<int,double> mypair(regIds[i],0.0);
    cumSubstitutedEnergy.insert(mypair);
    cumSubstitutedMaterial.insert(mypair);
    cumEmittedForOper.insert(mypair);
  }
}

void
Carbon::initialiseDeathBiomassStocks(const vector<double> & deathByFt, const int & regId){
  // it must initialize in the past the death biomass taking the value of the first year
  vector <string> fTypes = MTHREAD->MD->getForTypeIds();
  if(fTypes.size() != deathByFt.size()) {msgOut(MSG_CRITICAL_ERROR,"deathByFt and fTypes have different lenght!");}
  int currentYear = MTHREAD->SCD->getYear();
  //int initialYear = MD->getIntSetting("initialYear");

  for(uint i=0;i<fTypes.size();i++){
//    double life_inventory     =
 MTHREAD->MD->getForData("avgLive_deathBiomass_inventory",regId,fTypes[i],"",DATA_NOW);
//    double life_extra           =
 MTHREAD->MD->getForData("avgLive_deathBiomass_extra",regId,fTypes[i],"",DATA_NOW);
      double extraBiomass_ratio = MTHREAD->MD->getForData("extraBiomass_ratio",regId,
fTypes[i],"",DATA_NOW);

//    for(int y=currentYear;y>currentYear-life_inventory;y--){
//        iiskey key(y,regId,fTypes[i]);
//        pair<iiskey,double> mypair(key,deathByFt.at(i));
//        deathBiomassInventory.insert(mypair);
//    }
//    for(int y=currentYear;y>currentYear-life_extra;y--){
//        iiskey key(y,regId,fTypes[i]);
//        pair<iiskey,double> mypair(key,deathByFt.at(i)*extraBiomass_ratio);
//        deathBiomassExtra.insert(mypair);
//    }

    for(int y=currentYear;y>currentYear-100;y--){
        iiskey key(y,regId,fTypes[i]);
        pair<iiskey,double> mypairInventory(key,deathByFt.at(i));
        pair<iiskey,double> mypairExtra(key,deathByFt.at(i)*extraBiomass_ratio);
        deathBiomassInventory.insert(mypairInventory);
        deathBiomassExtra.insert(mypairExtra);
    }
  }
}

void
Carbon::initialiseProductsStocks(const vector<double> & qByProduct, const
int & regId){
  // it must initialize in the past the products taking the value of the first year
  vector <string> priProducts = MTHREAD->MD->getStringVectorSetting("
priProducts");
  vector <string> secProducts = MTHREAD->MD->getStringVectorSetting("
secProducts");
  vector <string> allProducts = priProducts;
  allProducts.insert( allProducts.end(), secProducts.begin(), secProducts.end() );
  if(allProducts.size() != qByProduct.size()) {msgOut(MSG_CRITICAL_ERROR,"
allProducts and qByProduct have different lenght!");}
  int currentYear = MTHREAD->SCD->getYear();
  for(uint i=0;i<allProducts.size();i++){
    double life  = MTHREAD->MD->getProdData("avgLife_products",regId,allProducts[i],
DATA_NOW);
    //for(int y=currentYear;y>currentYear-life;y--){
    for(int y=currentYear;y>currentYear-100;y--){
        iiskey key(y,regId,allProducts[i]);
        pair<iiskey,double> mypair(key,qByProduct.at(i));
        products.insert(mypair);
    }
  }
  //cout << "" << endl;
}

// ################ REGISTER FUNCTIONS ###############
void
Carbon::registerHarvesting(const double & value, const int & regId, const string
& fType){
  double convCoeff = MTHREAD->MD->getForData("forOperEmissions",regId,fType,""); //  Kg
 of CO2 emitted per cubic meter of forest operations
  // units:
  // value: Mm^3
  // convCoeff: Kg CO2/m^3 wood
  // desidered output: Mt CO2
  // ==> I must divide by 1000
  addSavedEmissions(-convCoeff*value/1000,regId,EM_FOROP);
  // Add the extraBiomass associated to the harvested volumes to the deathBiomassExtra pool
  int    year               = MTHREAD->SCD->getYear();
  double extraBiomass_ratio = MTHREAD->MD->getForData("extraBiomass_ratio",regId,fType,"
",DATA_NOW);
  double newDeathBiomass = value*extraBiomass_ratio;
  iiskey key(year,regId,fType);
  incrOrAddMapValue(deathBiomassExtra, key, newDeathBiomass);
}


void
Carbon::registerDeathBiomass(const double &value, const int & regId, const
string & fType){
  int year = MTHREAD->SCD->getYear();
  iiskey key(year,regId,fType);
  double extraBiomass_ratio = MTHREAD->MD->getForData("extraBiomass_ratio",regId,fType,"
",DATA_NOW);
  //pair<iiskey,double> mypairInventory(key,value);
  //pair<iiskey,double> mypairExtra(key,value*extraBiomass_ratio);
  incrOrAddMapValue(deathBiomassInventory, key, value);
  incrOrAddMapValue(deathBiomassExtra, key, value*extraBiomass_ratio);
  //deathBiomassInventory.insert(mypairInventory);
  //deathBiomassExtra.insert(mypairExtra);

}

void
Carbon::registerProducts(const double &value, const int & regId, const string & productName){
  // Registering the CO2 stock embedded in the product...
  int year = MTHREAD->SCD->getYear();
  iiskey key(year,regId,productName);
  pair<iiskey,double> mypair(key,value);
  products.insert(mypair);
  // registering the substituted CO2 for energy and material..
  double subEnergyCoeff   = MTHREAD->MD->getProdData("co2sub_energy",regId,productName,DATA_NOW,"");
  double subMaterialCoeff = MTHREAD->MD->getProdData("co2sub_material",regId,productName,DATA_NOW,"");
  // units:
  // value: Mm^3
  // subEnergyCoeff and subMaterialCoeff:  [kgCO2/m^3 wood]
  // desidered output: Mt CO2
  // ==> I must divide by 1000
  //addSavedEmissions(subEnergyCoeff*value/1000,regId,EM_ENSUB);
  addSavedEmissions(subMaterialCoeff*value/1000,regId,EM_MATSUB);
}



void
Carbon::registerTransports(const double &distQ, const int & regId){
  // units:
  // distQ: km*Mm^3
  // transportEmissionsCoeff:  [Kg CO2 / (km*m^3) ]
  // desidered output: Mt CO2
  // ==> I must divide by 1000
  double transportEmissionsCoeff = MTHREAD->MD->getDoubleSetting("transportEmissionsCoeff");
  addSavedEmissions(-transportEmissionsCoeff*distQ/1000,regId,EM_FOROP);
}

void
Carbon::HWP_eol2energy(){

  int currentYear = MTHREAD->SCD->getYear();
  int initialYear = MTHREAD->MD->getIntSetting("initialYear");
  vector <string> priProducts = MTHREAD->MD->getStringVectorSetting("priProducts");
  vector <string> secProducts = MTHREAD->MD->getStringVectorSetting("secProducts");
  vector <string> allProducts = priProducts;
  allProducts.insert( allProducts.end(), secProducts.begin(), secProducts.end() );

  vector<int> regIds = MTHREAD->MD->getRegionIds(2);
  for (uint r=0;r<regIds.size();r++){
    double regId = regIds[r];
    for(uint i=0;i<allProducts.size();i++){
      string pr = allProducts[i];
      double life   = MTHREAD->MD->getProdData("avgLife_products",regId,pr,DATA_NOW,"");
      double eol2e_share      = MTHREAD->MD->getProdData("eol2e_share",regId,pr,DATA_NOW,"");
      double subEnergyCoeff    = MTHREAD->MD->getProdData("co2sub_energy",regId,pr,DATA_NOW,"");
      if(eol2e_share > 0 && subEnergyCoeff>0){
        for(int y=(initialYear-100);y<currentYear;y++){ // notice the minor operator and not minor equal: energy substitution for products produced this year assigned to the following year, otherwise double counring in the process of making dicrete the exponential function
          iiskey key(y,regId,pr);
          double originalStock = findMap(products,key,MSG_NO_MSG,0.0);
          double remainingStockLastYear = getRemainingStock(originalStock,life,currentYear-y-1);
          double remainingStockThisYear = getRemainingStock(originalStock,life,currentYear-y);
          double eofThisYear = remainingStockLastYear-remainingStockThisYear;
          addSavedEmissions(subEnergyCoeff*eofThisYear*eol2e_share/1000,regId,EM_ENSUB);
        }
      }
    }
  }

}


// ################ UTILITY (PRIVATE) FUNCTIONS ###############

```

```
void
Carbon::addSavedEmissions(const double & value, const int & regId, const int & em_type){
  switch (em_type){
    case EM_ENSUB:
      incrMapValue(cumSubstitutedEnergy, regId, value);
      break;
    case EM_MATSUB:
      incrMapValue(cumSubstitutedMaterial, regId, value);
      break;
    case EM_FOROP:
      incrMapValue(cumEmittedForOper, regId, -value);
      break;
    default:
      msgOut(MSG_CRITICAL_ERROR,"Unexpected em_type in function getCumSavedEmissions");
  }
}

double
Carbon::getRemainingStock(const double & initialValue, const double & halfLife, const double & years) const{
  // // TODO: remove this test
  //if(years>0) return 0.0;
  //return initialValue;

  double k  = log(2)/halfLife;
  return initialValue*exp(-k*years);
}

```

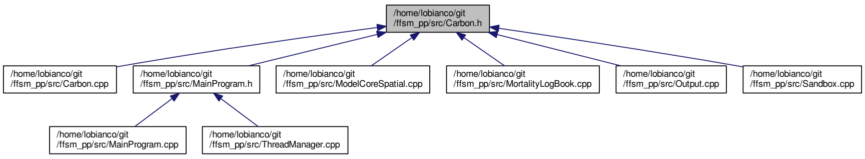
5.49 /home/lobianco/git/ffsm_pp/src/Carbon.h File Reference

```
#include <string>
#include <vector>
#include <map>
#include <stdexcept>
#include <iostream>
#include <sstream>
#include <time.h>
#include "BaseClass.h"
```

Include dependency graph for Carbon.h:



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



Classes

- class Carbon

  Class responsable to keep the logbook of the Carbon Balance.

5.50 Carbon.h

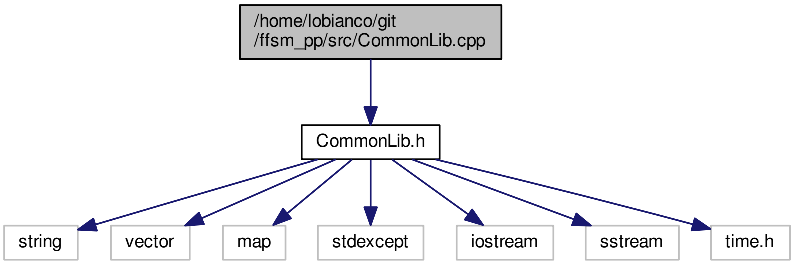
```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#ifndef CARBON_H
#define CARBON_H

// Core C++ headers
#include <string>
#include <vector>
#include <map>
#include <stdexcept>
#include <iostream>
#include <sstream>
#include <time.h>

//regmas headers
#include "BaseClass.h"

/// Class responsable to keep the logbook of the Carbon Balance
/**
@author Antonello Lobianco

A single istance of this class exists and is available trought the global MTHREAD->CBAL pointer.

It consits of functions to track a carbon-related event and store the information in STL maps that either register the events (for the stocks) or contain the cumulated carbon (for emission flows).

Carbon pools are stored as Mm^3 wood while and emission cumulated counters are directly in Mt CO2.

getStock() and getCumSavedEmissions() are then used to report the current levels of carbon in the stock or emitted/substituted.
*/

class Carbon: public BaseClass{
public:
                        Carbon(ThreadManager* MTHREAD_h); ///< Constructor
                       ~Carbon();


  double                getStock(const int & regId, const int & stock_type) const;       ///< Returns the current stock of carbon [Mt CO2]
  double                getCumSavedEmissions(const int & regId, const int & em_type) const;                 ///< Returns the current cumulative saved emissions by type [Mt CO2]

  void                  registerHarvesting(const double & value, const int & regId, const string &fType);   ///< Registers the harvesting of trees increasing the value of cumEmittedForOper
  void                  registerDeathBiomass(const double &value, const int & regId, const string &fType);  ///< Registers the "death" of a given amount of biomass, storing it in the deathBiomass map
  void                  registerProducts(const double &value, const int & regId, const string &productName);///< Registers the production of a given amount of products, storing it in the products maps. Also increase material substitution.
  void                  registerTransports(const double &distQ, const int & regId);                   ///< Registers the quantities emitted by transport of wood FROM a given region
  void                  initialiseDeathBiomassStocks(const vector<double> & deathByFt, const int & regId);  ///< Initialises the stocks of death biomass for the avgLive_* years before the
```

```
       simulation starts
        void                    initialiseProductsStocks(const vector<double> & qByProduct,
       const int &regId);        ///< Initialises the stocks of products for the avgLive_* years before the
       simulation starts
        void                    initialiseEmissionCounters();
                                   ///< Initialises the emission counters to zero.
        void                    HWP_eol2energy();
                      ///< Computes the energy substitution for the quota of HWP that reachs end of life and
       doesn’t go to landfill


      private:
        void                    addSavedEmissions(const double & value, const int & regId, const
      int & em_type);    ///< Increases the value to the saved emissions for a given type and region
        double                  getRemainingStock(const double & initialValue, const double &
      halfLife, const double & years) const; ///< Apply a single exponential decay model to retrieve the remining
       stock given the initial stock, the half life and the time passed from stock formation.

        map<iiskey, double >     deathBiomassInventory; ///< Map that register the death of
       biomass by year, l2_region and forest type (inventoried)[Mm^3 wood]
        map<iiskey, double >         deathBiomassExtra; ///< Map that register the death of
       biomass by year, l2_region and forest type (non-inventoried biomass: branches, roots..) [Mm^3 wood]
        map<iiskey, double >                  products; ///< Map that register the production of a given
       product by year, l2_region and product [Mm^3 wood]
        map<int,double>            cumSubstitutedEnergy; ///< Map that store the cumulative
       CO2 substituted for energy consumption, by l2_region [Mt CO2]
        map<int,double>          cumSubstitutedMaterial; ///< Map that store the cumulative
       CO2 substituted using less energivory material, by l2_region [Mt CO2]
        map<int,double>              cumEmittedForOper; ///< Map that store emissions for forest
       operations, including transport, by l2_region [Mt CO2]



      };

      #endif // CARBON_H
```

5.51 /home/lobianco/git/ffsm_pp/src/CommonLib.cpp File Reference

`#include "CommonLib.h"`

Include dependency graph for CommonLib.cpp:



Functions

- double testB ()

5.51.1 Function Documentation

5.51.1.1 double testB ( )

Definition at line 40 of file CommonLib.cpp.

```
00040                    {
00041  return 2.0;
00042 }
```

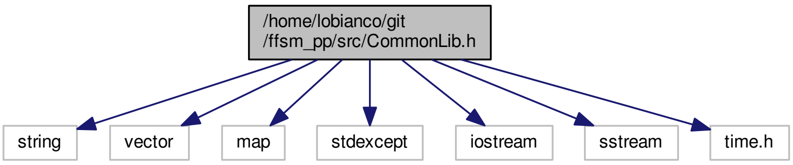
5.52 CommonLib.cpp

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d'Economie Forestière              *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify *
 *   it under the terms of the GNU General Public License as published by *
 *   the Free Software Foundation; either version 3 of the License, or    *
 *   (at your option) any later version, given the compliance with the    *
 *   exceptions listed in the file COPYING that is distribued together    *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,      *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of       *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the        *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License    *
 *   along with this program; if not, write to the                        *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.            *
 ***************************************************************************/

#include "CommonLib.h"




/*template <typename K>  int getMaxPosition (const vector <K> & aVector){
   return 0;
}*/

/*template <typename K> int getMaxPosition (const vector <k> & aVector){
   return 0;
}

template int getMaxPosition<int>(const vector <int> & aVector);
template int getMaxPosition<double>(const vector <double> & aVector);*/


double testB (){
 return 2.0;
}

```

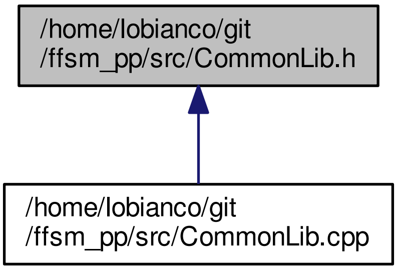
5.53 /home/lobianco/git/ffsm_pp/src/CommonLib.h File Reference

```
#include <string>
#include <vector>
#include <map>
#include <stdexcept>
#include <iostream>
#include <sstream>
#include <time.h>
```

Include dependency graph for CommonLib.h:



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



Functions

- double testB ()

5.53.1 Function Documentation

5.53.1.1 double testB ( )

Definition at line 40 of file CommonLib.cpp.

```
00040                 {
00041  return 2.0;
00042 }
```

5.54 CommonLib.h

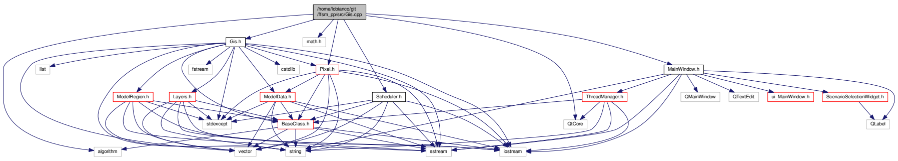
```
/****************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière                *
 *   http://ffsm-project.org                                                *
 *                                                                          *
 *   This program is free software; you can redistribute it and/or modify   *
 *   it under the terms of the GNU General Public License as published by   *
 *   the Free Software Foundation; either version 3 of the License, or      *
 *   (at your option) any later version, given the compliance with the      *
 *   exceptions listed in the file COPYING that is distribued together      *
 *   with this file.                                                        *
 *                                                                          *
 *   This program is distributed in the hope that it will be useful,        *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of         *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the          *
 *   GNU General Public License for more details.                           *
 *                                                                          *
 *   You should have received a copy of the GNU General Public License      *
 *   along with this program; if not, write to the                          *
 *   Free Software Foundation, Inc.,                                        *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.              *
 ***************************************************************************/
#ifndef COMMONLIB_H
#define COMMONLIB_H

// Core C++ headers
#include <string>
#include <vector>
#include <map>
```

```
#include <stdexcept>
#include <iostream>
#include <sstream>
#include <time.h>

// regmas headers...


/**
A file containing utility functions
*/

using namespace std;

/*template <typename K>  int getMaxPosition (const vector <K> & aVector);*/
//template <typename K> int getMaxPosition (const vector <K> & aVector);
//int testAA (const vector <double> & atest){return 1;}
double testB ();






#endif // CARBON_H
```

5.55 /home/lobianco/git/ffsm_pp/src/Gis.cpp File Reference

```
#include <algorithm>
#include <QtCore>
#include <math.h>
#include "Gis.h"
#include "Pixel.h"
#include "MainWindow.h"
#include "Scheduler.h"
```

Include dependency graph for Gis.cpp:



5.56 Gis.cpp

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d'Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify *
 *   it under the terms of the GNU General Public License as published by *
 *   the Free Software Foundation; either version 3 of the License, or    *
 *   (at your option) any later version, given the compliance with the    *
 *   exceptions listed in the file COPYING that is distribued together    *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,      *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of       *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the        *
 *   GNU General Public License for more details.                         *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License    *
 *   along with this program; if not, write to the                        *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#include <algorithm> //alghoritm used to shuffle (randomize) the array
#include <QtCore>
```

```
#include <math.h>

#include "Gis.h"
#include "Pixel.h"

//#include "InputDocument.h"
#include "MainWindow.h"
#include "Scheduler.h"

using namespace std;

/**
The constructor of the GIS (unique) instance want:
@param RD_h      Pointer to the manager of the regional data
@param MTHREAD_h Pointer to the main thread manager
*/
Gis::Gis(ThreadManager* MTHREAD_h){
  MTHREAD=MTHREAD_h;
}

Gis::~Gis(){
}

/**
setSpace is called directly from the init system to setting the space environment in the model.
<br>It is responsable to:
 - define map dimensions (from setting files)
 - create the pixels
 - initialize the layer @see initLayers
 - load the layer data from their fdata-files @see loadLayersDataFromFile
 - tell the GUI that our map will have (x,y) dimensions
*/
void
Gis::setSpace(){



  msgOut(MSG_INFO,"Creating the space...");

  // init basic settings....
  geoTopY = MTHREAD->MD->getDoubleSetting("geoNorthEdge");
  geoBottomY = MTHREAD->MD->getDoubleSetting("geoSouthEdge");
  geoLeftX = MTHREAD->MD->getDoubleSetting("geoWestEdge");
  geoRightX = MTHREAD->MD->getDoubleSetting("geoEastEdge");
  xNPixels = MTHREAD->MD->getIntSetting("nCols");
  yNPixels = MTHREAD->MD->getIntSetting("nRows");
  noValue = MTHREAD->MD->getDoubleSetting("noValue");
  xyNPixels = xNPixels * yNPixels;
  xMetersByPixel = (geoRightX - geoLeftX)/xNPixels;
  yMetersByPixel = (geoTopY - geoBottomY)/yNPixels;
  MTHREAD->treeViewerChangeGeneralPropertyValue("total plots", d2s(getXyNPixels()));
  MTHREAD->treeViewerChangeGeneralPropertyValue("total land", d2s(xyNPixels*getHaByPixel()));
  // creating pixels...
  for (int i=0;i<yNPixels;i++){
    for (int j=0;j<xNPixels;j++){
      Pixel myPixel(i*xNPixels+j, MTHREAD);
      myPixel.setCoordinates(j,i);
      pxVector.push_back(myPixel);
    }
  }
  initLayers();
  loadLayersDataFromFile();

  // Cashing the pixels owned by each region..
  vector <ModelRegion*> regions = MTHREAD->MD->getAllRegions();
  int nRegions = regions.size();
  for(uint i=0;i<nRegions;i++){
    regions[i]->setMyPixels();
  }

  applySpatialStochasticValues(); // regional variance -> different tp in each pixel trought tp modifiers
  applyStochasticRiskAdversion(); // risk adversion to each pixel
  cachePixelValues(); // For computational reasons cache some values in the constant layers directly as properties of the pixel object

//  //< Print a layer of pixels id..
//  addLayer("pxIds", "idx of the pixels", true, true, "pxIds.grd", true);
//  resetLayer("pxIds");
//  vector<Pixel*> allPixels = getAllPlotsByRegion(11000);
//  for (int i=0;i<allPixels.size();i++){
//    int pxId= allPixels[i]->getID();
//    allPixels[i]->changeValue ("pxIds", pxId);
//  }
//  printLayers("pxIds");


  MTHREAD->fitInWindow(); // tell the gui to fit the map to the widget
```

```
//  countItems("landUse",false); // count the various records assigned to each legendItem. Do not print debug infos
  return;
}


/**
 * Apply all stochastic modifications required by the model at init time.
 * Currently used to change time of passage devending on regional variance
 **/

void
Gis::applySpatialStochasticValues(){
  // apply regional volume growth st.dev. -> variance to pixel based t.p.
  // - cashing value to the pixels
  // - apply to the tp layers with change values

  if(!MTHREAD->MD->getBoolSetting("usePixelData")) return;

  vector <int> regIds2 = MTHREAD->MD->getRegionIds(2);
  //ModelRegion* reg = MTHREAD->MD->getRegion(regIds2[i]);
  //vector<Pixel*> regPixels = region->getMyPixels();
  //double sumc = 0;
  //double nc = 0;
  for(uint i=0;i<regIds2.size();i++){
    ModelRegion* reg = MTHREAD->MD->getRegion(regIds2[i]);
    vector <Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(regIds2[i]);
    vector <string> fTypes = MTHREAD->MD->getForTypeIds();

    // regional variance
    if(MTHREAD->MD->getBoolSetting("useSpatialRegionalVariance")){
      for(uint j=0; j<fTypes.size(); j++){
        double sStDev = MTHREAD->MD->getForData("sStDev",regIds2[i],fTypes[j],""); // spatial standard deviation
        double agr = MTHREAD->MD->getForData("agr",regIds2[i],fTypes[j],""); // average growth
        // BUG solved 20141220 To obtain a population with the same avg and st.dev of the original using moltipliers, I need to use the cv not the st.dev. !
        // tested with excel
        normal_distribution<double> d(1,sStDev/agr); // default any how to double
        for (uint z=0;z<rpx.size();z++){
          double c = d(*MTHREAD->gen);
          double c2 = max(0.4,min(1.6,c)); /// with simmetric boundary on the cv I do not change the average, but of course I slighly reduce the stdev. See file monte_carlo_with_multipliers_sample_proof.ods
          // TO.DO: Convert it to using normSample where instead of a min/max a loop is used to fund smaples that are within the bounds
          //cout << regIds2[i] << ";" <<sStDev <<";"<< c <<endl
          //rpx[z]->correctInputMultiplier("tp_multiplier",fTypes[j],c);
          //cout << sStDev/agr << "    " << c2 << endl;
          rpx[z]->setSpModifier(c2,j);
          //sumc += c;
          //nc ++;
        }
      }
    }

    // expectation types
    double avgExpTypes       = MTHREAD->MD->getDoubleSetting("expType");
    double avgExpTypesPrices = MTHREAD->MD->getDoubleSetting("expTypePrices");
    double expTypes_cv       = MTHREAD->MD->getDoubleSetting("expType_cv");
    double expTypesPrices_cv = MTHREAD->MD->getDoubleSetting("expTypePrices_cv");
    if((avgExpTypes<0 || avgExpTypes>1) && avgExpTypes != -1){
      msgOut(MSG_CRITICAL_ERROR, "expType parameter must be between 1 (expectations) and 0 (adaptative) or -1 (fixed).");
    }
    if(avgExpTypesPrices<0 || avgExpTypesPrices>1){
      msgOut(MSG_CRITICAL_ERROR, "vgExpTypesPrices parameter must be between 1 (expectations) and 0 (adaptative).");
    }
    //cout << avgExpTypes << "     " << expTypes_cv << endl;

    normal_distribution<double> exp_distr(avgExpTypes,expTypes_cv *avgExpTypes); // works only for double, but default any how to double
    normal_distribution<double> expPrices_distr(avgExpTypesPrices,expTypesPrices_cv *avgExpTypesPrices);

    for (uint z=0;z<rpx.size();z++){
      if(avgExpTypes == -1){
        rpx[z]->expType = -1;
      } else {
        //double c = exp_distr(*MTHREAD->gen);
        //double c2 = max(0.0,min(1.0,c)); /// Bounded [0,1]. With simmetric boundary on the cv I do not change the average, but of course I slighly reduce the stdev. See file monte_carlo_with_multipliers_sample_proof.ods
        double c3 = normSample(exp_distr,*MTHREAD->gen,0.0,1.0);
        //cout << "Sampled:\t" << c3 <<  endl;
        rpx[z]->expType = c3;
      }
      double cPrice = normSample(expPrices_distr,*MTHREAD->gen,0.0,1.0);
      rpx[z]->expTypePrices = cPrice;
    }
  }
}

/**
 * Apply to each agent a random risk-adversion coefficient
 *
 *For now, 1 pixel = 1 agent, and avg and st.dev. are the same in the model, but eventually this can change
 **/
void
Gis::applyStochasticRiskAdversion(){
  // apply regional volume growth st.dev. -> variance to pixel based t.p.
  // - cashing value to the pixels
  // - apply to the tp layers with change values

  if(!MTHREAD->MD->getBoolSetting("usePixelData")) return;

  vector <int> regIds2 = MTHREAD->MD->getRegionIds(2);
  bool raEnabled = MTHREAD->MD->getBoolSetting("heterogeneousRiskAversion");
  for(uint i=0;i<regIds2.size();i++){
    ModelRegion* reg = MTHREAD->MD->getRegion(regIds2[i]);
    vector <Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(regIds2[i]);
    for (uint z=0;z<rpx.size();z++){
      if(raEnabled){
        double raStDev = MTHREAD->MD->getDoubleSetting("riskAversionAgentSd");
        double avg = MTHREAD->MD->getDoubleSetting("riskAversionAgentAverage");
        normal_distribution<double> d(avg,raStDev); // default any how to double
        double c = d(*MTHREAD->gen);
        rpx[z]->setValue ("ra", c);
      } else {
        rpx[z]->setValue ("ra", 0.0);
      }
    }
  }
}

void
Gis::cachePixelValues(){
  /// Set the avalCoef (availability coefficient) from layer
  if(!MTHREAD->MD->getBoolSetting("usePixelData")) return;

  bool applyAvalCoef = MTHREAD->MD->getBoolSetting("applyAvalCoef");
  vector <int> regIds2 = MTHREAD->MD->getRegionIds(2);

  for(uint i=0;i<regIds2.size();i++){
    ModelRegion* reg = MTHREAD->MD->getRegion(regIds2[i]);
    vector <Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(regIds2[i]);
    for (uint p=0;p<rpx.size();p++){
      if(applyAvalCoef){
        rpx[p]->avalCoef = rpx[p]->getDoubleValue("avalCoef", true);
      }
    }
  }
}

/**
Called from setSpace(), initLayers() is responsable of:
 - load each layer propriety (name, label, datafile..)
 - add the layer to the system @see addLayer
<p>If the layer is to be read at start-up:
 - adding to the layer each legend item (ID, label, min-max values..) @see addLegendItem
 - [REMOVED, as reclassification rules are in the input ods file now, not in the gis input file] eventually adding to the layer each reclassification rules @see addReclassificationRule
**/
void
Gis::initLayers(){
  // setting layers...
  //string filename_complete= MTHREAD->MD->getFilenameByType("gis");
  string filename_complete = MTHREAD->getBaseDirectory()+MTHREAD->MD->getStringSetting("gisFilename");

  InputNode gisDocument;
  bool test=gisDocument.setWorkingFile(filename_complete);
  if (!test){msgOut(MSG_CRITICAL_ERROR, "Error opening the gis file "+filename_complete+".");}
  vector<InputNode> layerNodes = gisDocument.getNodesByName("layer");
  vector<string> ftIds = MTHREAD->MD->getForTypeIds();
  for (uint i=0; i<layerNodes.size();i++){

    string nameOrig = layerNodes.at(i).getNodeByName("name").getStringContent();
    string labelOrig = layerNodes.at(i).getNodeByName("label").getStringContent();
    bool isInteger = layerNodes.at(i).getNodeByName("isInteger").getBoolContent();
    bool dynamicContent = layerNodes.at(i).getNodeByName("dynamicContent").getBoolContent();
    bool expandByFt = layerNodes.at(i).getNodeByName("expandByFt").getBoolContent();
    string readAtStart = layerNodes.at(i).getNodeByName("readAtStart").getStringContent();
    if (readAtStart != "true") continue;
    string dirName = layerNodes.at(i).getNodeByName("dirName").getStringContent();
  string fileName = layerNodes.at(i).getNodeByName("fileName").getStringContent();

  // Eventually expanding this input layern in as many layer as forest types exists..
  uint endingLoop = expandByFt ? ftIds.size(): 1;
  for(uint z=0;z<endingLoop;z++){
   string ftExtension= expandByFt ? "_"+ftIds[z]:"";
   string labelFtExtension= expandByFt ? " ("+ftIds[z]+")":"";
   string name = nameOrig+ftExtension;
   string label = labelOrig + labelFtExtension;

   string fullFileName = ((dirName == "") || (fileName==""))?"":MTHREAD->MD->getBaseDirectory()+dirName+fileName+ftExtension; // TODO: ugly: one would have to put mmyfile.grd_broadL_highF
   addLayer(name,label,isInteger,dynamicContent,fullFileName);
   //legend..
   vector<InputNode> legendItemsNodes = layerNodes.at(i).getNodesByName("legendItem");
   for (uint j=0; j<legendItemsNodes.size();j++){
    int lID = legendItemsNodes.at(j).getIntContent();
    string llabel = legendItemsNodes.at(j).getStringAttributeByName("label");
    int rColor = legendItemsNodes.at(j).getIntAttributeByName("rColor");
    int gColor = legendItemsNodes.at(j).getIntAttributeByName("gColor");
    int bColor = legendItemsNodes.at(j).getIntAttributeByName("bColor");
    double minValue, maxValue;
    if (isInteger){
     minValue = ((double)lID);
     maxValue = ((double)lID);
    }
    else {
     minValue = legendItemsNodes.at(j).getDoubleAttributeByName("minValue");
     maxValue = legendItemsNodes.at(j).getDoubleAttributeByName("maxValue");
    }
    addLegendItem(name, lID, llabel, rColor, gColor, bColor, minValue, maxValue);
   }
  }
 }
 initLayersPixelData();
 //initLayersModelData(DATA_INIT); // only the layers relative to the initial years are inserted now. All the simulation year layers will be added each year before mainSimulationyear()
}

/** Init the layers of exogenous data at pixel level (e.g. time of passage, multipliers, volumes of sp. espl. ft, spread models)
  These layers will then be read from datafile
*/
void
Gis::initLayersPixelData(){
 if (!MTHREAD->MD->getBoolSetting("usePixelData")){return;}
 string dir = MTHREAD->MD->getBaseDirectory()+MTHREAD->MD->getStringSetting("spatialDataSubfolder");
 string fileExt = MTHREAD->MD->getStringSetting("spatialDataFileExtension");
 vector<string> files = vector<string>();
 string fullFilename, filename, fullPath;
 //string parName, forName, dClass, yearString;
 //int year;

 MTHREAD->MD->getFilenamesByDir (dir,files, fileExt); // Ugly format. Files is the output (reference)

 for (unsigned int i = 0;i < files.size();i++) {
  fullFilename = files[i];
  fullPath = dir+"/"+fullFilename;
  filename = fullFilename.substr(0,fullFilename.find_last_of("."));
  addLayer(filename,filename,false,false,fullPath,false);
 }

 // Loading volumes of forest types that are spatially known..
 if(MTHREAD->MD->getBoolSetting("useSpExplicitForestTypes")){
  string dir2 = MTHREAD->MD->getBaseDirectory()+MTHREAD->MD->getStringSetting("spExplicitForTypesInputDir");
  string fileExt2 = MTHREAD->MD->getStringSetting("spExplicitForTypesFileExtension");
  vector<string> files2 = vector<string>();
  string fullFilename2, filename2, fullPath2;
  MTHREAD->MD->getFilenamesByDir (dir2,files2, fileExt2); // Ugly format. Files is the output (reference)
  for (unsigned int i = 0;i < files2.size();i++) {
   fullFilename2 = files2[i];
   fullPath2 = dir2+"/"+fullFilename2;
   filename2 = fullFilename2.substr(0,fullFilename2.find_last_of("."));
   addLayer(filename2,filename2,false,false,fullPath2,false);
  }
 }

 // Loading pathogens exogenous spread models...
 if(MTHREAD->MD->getBoolSetting("usePathogenModule")){
  string dir2 = MTHREAD->MD->getBaseDirectory()+MTHREAD->MD->getStringSetting("pathogenExogenousSpreadModelFolder");
  string fileExt2 = MTHREAD->MD->getStringSetting("pathogenExogenousSpreadModelFileExtension");
  vector<string> files2 = vector<string>();
  string fullFilename2, filename2, fullPath2;
  MTHREAD->MD->getFilenamesByDir (dir2,files2, fileExt2); // Ugly format. Files is the output (reference)
  for (unsigned int i = 0;i < files2.size();i++) {
      fullFilename2 = files2[i];
      fullPath2 = dir2+"/"+fullFilename2;
      filename2 = fullFilename2.substr(0,fullFilename2.find_last_of("."));
      addLayer(filename2,filename2,false,false,fullPath2,false);
    }
  }

}

/** Init the layers of exogenous data at pixel level (e.g. time of passage)
    These layers will NOT be read by datafile, but volume for each pixel will be calculated from regional data and area map
*/
/*
void
Gis::initLayersModelData(const int& year_h){

  if (!MTHREAD->MD->getBoolSetting("usePixelData")) return;

  vector <int> years;
  if(year_h==DATA_NOW){
    years.push_back(MTHREAD->SCD->getYear());
  } else if (year_h==DATA_INIT){
    int initialYear = MTHREAD->MD->getIntSetting("initialYear");
    int initialOptYear = MTHREAD->MD->getIntSetting("initialOptYear");
    for(int y=initialYear;y<initialOptYear;y++){
      years.push_back(y);
    }
  } else {
    years.push_back(year_h);
  }

  vector <string> dClasses = MTHREAD->MD->getStringVectorSetting("dClasses");
  vector <string> fTypes = MTHREAD->MD->getForTypeIds();
  //int initialYear = MTHREAD->MD->getIntSetting("initialYear");
  //int initialOptYear = MTHREAD->MD->getIntSetting("initialOptYear");
  //int simYears = MTHREAD->MD->getIntSetting("simulationYears");
  string layerName_vol,layerName_cumTp,layerName_regArea,layerName_area;
  for(uint i=0;i< fTypes.size();i++){
    for(int y=0;y<years.size();y++){
      layerName_regArea = pack("regArea",fTypes[i],"",years[y]);
      addLayer(layerName_regArea,layerName_regArea,false,true,"",false);
      for (uint j=0;j<dClasses.size();j++){
        layerName_vol = pack("vol",fTypes[i],dClasses[j],years[y]);
        layerName_cumTp = pack("cumTp",fTypes[i],dClasses[j],years[y]);
        layerName_area = pack("area",fTypes[i],dClasses[j],years[y]);
        addLayer(layerName_vol,layerName_vol,false,true,"",false);
        addLayer(layerName_cumTp,layerName_cumTp,false,true,"",false);
        addLayer(layerName_area,layerName_area,false,true,"",false);
      }

    }

  }
  string debug = "done";

}
*/

Layers*
Gis::getLayer(const string& layerName_h){
  for(uint i=0;i<layerVector.size();i++){
    if(layerVector[i].getName() == layerName_h){
      return &layerVector[i];
    }
  }
  msgOut(MSG_CRITICAL_ERROR, "Layer "+layerName_h+" not found. Aborting.");
}

void
Gis:: applyForestReclassification(){
/*per ogni forest type:
 - crea i layers delle forest type nuovi
 - riempi con zero
 - passa le info dal layerr ereditato al nuovo
 per ogni pixel
 */

  // caching
  int nReclassRules = MTHREAD->MD->getNReclRules();
  vector <reclRule*> RRs;
  for(uint z=0;z<nReclassRules;z++){
    RRs.push_back(MTHREAD->MD->getReclRule(z));
  }


  for (uint i=0;i< MTHREAD->MD->getNForTypes();i++){
    forType* FT = MTHREAD->MD->getForType(i);
    if(!layerExist(FT->forLayer)){
      addLayer(FT->forLayer, "Are layer for forest type "+FT->forTypeId, false, true);
      resetLayer(FT->forLayer);
      Layers* newLayer = getLayer(FT->forLayer);
      Layers* ereditatedLayer = getLayer(MTHREAD->MD->getForType(FT->ereditatedFrom)->forLayer);
      newLayer->addLegendItems(ereditatedLayer->getLegendItems());
    }
  }


  for (uint i=0;i< MTHREAD->MD->getNForTypes();i++){
    forType* FT = MTHREAD->MD->getForType(i);
    for(uint j=0;j<xyNPixels;j++){
      Pixel* PX = getPixel(j);
      //int regL1 =  PX->getDoubleValue ("regLev_1");
      int regL2 =  PX->getDoubleValue ("regLev_2");
      double value =  PX->getDoubleValue (FT->forLayer, true);
      for(uint z=0;z<nReclassRules;z++){
        reclRule* RR = RRs[z];
        //if( (RR->regId == regL2 || RR->regId == regL1) && RR->forTypeOut == FT->forTypeId ){ // we found a reclassification rule for the region where is located this pixel and that output on the for type we are using
        if( RR->regId == regL2  && RR->forTypeOut == FT->forTypeId ){ // we found a reclassification rule for the region where is located this pixel and that output on the for type we are using
          string debugForTypeIn = RR->forTypeIn;
          double inputValue = PX->getDoubleValue(MTHREAD->MD->getForType(RR->forTypeIn)->forLayer, true);
          double reclassCoeff = RR->coeff;
          value += inputValue * reclassCoeff ;
          // not breaking because we may have more than one input for the same output
        }
      }
      PX->changeValue(FT->forLayer, value, true);
    }
    updateImage(FT->forLayer);
  }
  //countItems("forType_B_HF", true);
  refreshGUI();
  /*Pixel* DP = getPixel(8386);
  msgOut(MSG_DEBUG,"Debug info on plot 8386");
  for (uint i=0;i< MTHREAD->MD->getNForTypes();i++){
    forType* FT = MTHREAD->MD->getForType(i);
    msgOut(MSG_DEBUG,FT->forTypeId+" - "+d2s(DP->getDoubleValue (FT->forLayer)));
  }
  */
}


/**
Called at init time from initLayers, or during model run-time, this function will add a layer to the system.
@param name_h ID of the layer (no spaces!)
@param label_h layer label
@param type_h type of the layer, integer or contiguous
@param dynamicContent_h if it change during the time (so it needs to be printed each year) or not
@param fullFilename_h if the layer has to be read at the beginning, the name of the associated datafile (default="")
<p>It:
 - had the layer to the layerVector
 - set all pixels with nodata for that specific layer
 - let the GUI know we have a new layer
*/
void
Gis::addLayer(string name_h, string label_h, bool isInteger_h, bool dynamicContent_h, string fullFileName_h, bool display_h){
  if(name_h == "forArea_ash"){
    bool debug = true;
  }
  for(uint i=0; i<layerVector.size(); i++){
    if (layerVector.at(i).getName() == name_h){
      msgOut(MSG_ERROR, "Layer already exist with that name");
      return;
    }
  }
  Layers LAYER (MTHREAD, name_h, label_h, isInteger_h, dynamicContent_h, fullFileName_h, display_h);
  layerVector.push_back(LAYER);

  for (uint i=0;i<xyNPixels; i++){
    pxVector[i].setValue(name_h,noValue);
  }
  if(display_h){
    MTHREAD->addLayer(name_h,label_h);
  }

}

void
Gis::resetLayer(string layerName_h){

  for(uint i=0; i<layerVector.size(); i++){
    if (layerVector.at(i).getName() == layerName_h){
      for (uint i=0;i<xyNPixels; i++){
        pxVector.at(i).changeValue(layerName_h,noValue); // bug solved 20071022, Antonello
      }
      return;
    }
  }
  msgOut(MSG_ERROR, "I could not reset layer "+layerName_h+" as it doesn’t exist!");
}

bool
Gis::layerExist(const string& layerName_h, bool exactMatch) const{

  if(exactMatch){
    for(uint i=0; i<layerVector.size(); i++){
      if (layerVector.at(i).getName() == layerName_h){
        return true;
      }
    }
  } else { // partial matching (stored layer name begin with search parameter)
    for(uint i=0; i<layerVector.size(); i++){
      if (layerVector.at(i).getName().compare(0, layerName_h.size(),layerName_h )){
        return true;
      }
    }
  }

  return false;
}

/**
Search within the layerVector and call addLegendItem(...) to the appropriate one.
<p>Called at init time from initLayers, or during model run-time.
@param name_h Name of the layer
@param ID_h ID of the specific lagend item
@see Layers::addLegendItem
*/
void
Gis::addLegendItem(string name_h, int ID_h, string label_h, int rColor_h, int gColor_h, int bColor_h, double minValue_h, double maxValue_h){

  for(uint i=0; i<layerVector.size(); i++){
    if (layerVector.at(i).getName() == name_h){
      layerVector.at(i).addLegendItem(ID_h, label_h, rColor_h, gColor_h, bColor_h, minValue_h, maxValue_h);
      return;
    }
  }
  msgOut(MSG_ERROR, "Trying to add a legend item to a layer that doesn’t exist.");
  return;
}

/**
Search within the layerVector and call countMyPixels(...) to the appropriate one.
<p>Called at init time from initLayers, or during model run-time.
@param layerName_h Name of the layer
@param debug Print the values on the GUI
@see Layers::countMyPixels
*/
void
Gis::countItems(const string &layerName_h, const bool &debug){

  for(uint i=0; i<layerVector.size(); i++){
    if (layerVector.at(i).getName() == layerName_h){
      layerVector.at(i).countMyPixels(debug);
      return;
    }
  }
  msgOut(MSG_ERROR, "Trying to get statistics (count pixels) of a layer that doesn’t exist.");
  return;
}


/**
Called at init time from initLayers, this function load the associated datafile to the existing layers (that if exists at this stage are all of type to be loaded at start-up).
<br>This function loop over layerVector and works with GRASS/ASCII (tested) or ARC/ASCII (untested) datasets, assigning to each pixel the readed value to the corresponding layer.
<br>The function also "compose" the initial map with the colors read by the layer (for each specific values) and send the map to the GUI.

```

```
NOTE: It uses some Qt functions!!!

@see Pixel::changeValue
@see Layers::filterExogenousDataset
@see Layers::getColor
*/
void
Gis::loadLayersDataFromFile(){
  double localNoValue = noValue;
  double inputValue;
  double outputValue;
  QColor color;

  for(uint i=0;i<layerVector.size();i++){
    string layerName =layerVector.at(i).getName();
    string fileName=layerVector.at(i).getFilename();
    if(fileName == "") continue; // BUGGED !!! 20121017, Antonello. It was "return", so it wasn't reading any layers following a layer with no filename
    QFile file(fileName.c_str());
    if (!file.open(QFile::ReadOnly)) {
      cerr << "Cannot open file for reading: "
        << qPrintable(file.errorString()) << endl;
      msgOut(MSG_ERROR, "Cannot open map file "+fileName+" for reading.");
      continue;
    }
    QTextStream in(&file);
    int countRow = 0;
    QImage image = QImage(xNPixels, yNPixels, QImage::Format_RGB32);
    image.fill(qRgb(255, 255, 255));
    while (!in.atEnd()) {
      QString line = in.readLine();
      QStringList fields = line.split(' ');
      if (
        (fields.at(0)== "north:" && fields.at(1).toDouble() != geoTopY)
        || ((fields.at(0)== "south:" || fields.at(0) == "yllcorner" ) && fields.at(1).toDouble() != geoBottomY)
        || (fields.at(0)== "east:" && fields.at(1).toDouble() != geoRightX)
        || ((fields.at(0)== "west:" || fields.at(0) == "xllcorner" ) && fields.at(1).toDouble() != geoLeftX)
        || ((fields.at(0)== "rows:" || fields.at(0) == "nrows" ) && fields.at(1).toInt() != yNPixels)
        || ((fields.at(0)== "cols:" || fields.at(0) == "ncols" ) && fields.at(1).toInt() != xNPixels)
      )
      {
        msgOut(MSG_ERROR, "Layer "+layerName+" has different coordinates. Aborting reading.");
        break;
      } else if (fields.at(0)== "null:" || fields.at(0) == "NODATA_value" || fields.at(0) == "nodata_value" ) {
        localNoValue = fields.at(1).toDouble();
      } else if (fields.size() > 5) {
        for (int countColumn=0;countColumn<xNPixels;countColumn++){
          inputValue = fields.at(countColumn).toDouble();
          if (inputValue == localNoValue){
            outputValue = noValue;
            pxVector.at((countRow*xNPixels+countColumn)).changeValue(layerName,outputValue);
            QColor nocolor(255,255,255);
            color = nocolor;
          }
          else {
            outputValue=layerVector.at(i).filterExogenousDataset(fields.at(countColumn).toDouble());
            pxVector.at((countRow*xNPixels+countColumn)).changeValue(layerName,outputValue);
            color = layerVector.at(i).getColor(outputValue);
          }
          image.setPixel(countColumn,countRow,color.rgb());
        }
        countRow++;
      }
    }
    if (MTHREAD->MD->getBoolSetting("initialRandomShuffle") ){
      layerVector.at(i).randomShuffle();
    }
    this->filterSubRegion(layerName);
    if(layerVector.at(i).getDisplay()){
      MTHREAD->updateImage(layerName,image);
      //send the image to the gui...
      refreshGUI();
    }


  }
}

/**
Update an ALREADY EXISTING image and send the updated image to the GUI.
<br>It is used instead of updating the individual pixels that is much more time consuming than change the individual pixels value and then upgrade the image as a whole.
@param layername_h Layer from where get the image data
*/
```

```
void
Gis::updateImage(string layerName_h){
  msgOut (1, "Update image "+layerName_h+"...");

  // sub{X,Y}{R,L,T,B} refer to the subregion coordinates, but when this is not active they coincide with the whole region
  QImage image = QImage(subXR-subXL+1, subYB-subYT+1, QImage::Format_RGB32);

  image.fill(qRgb(255, 255, 255));
  int layerIndex=-1;
  for (uint i=0;i<layerVector.size();i++){
    if (layerVector.at(i).getName() == layerName_h){
      layerIndex=i;
      break;
    }
  }
  if (layerIndex <0) {
    msgOut(MSG_CRITICAL_ERROR, "Layer not found in Gis::updateImage()");
  }

  for (int countRow=subYT;countRow<subYB;countRow++){
    for (int countColumn=subXL;countColumn<subXR;countColumn++){
      double value = pxVector.at((countRow*xNPixels+countColumn)).getDoubleValue(layerName_h);
      QColor color = layerVector.at(layerIndex).getColor(value);
      image.setPixel(countColumn-subXL,countRow-subYT,color.rgb());
    }
  }
  MTHREAD->updateImage(layerName_h,image);
  refreshGUI();
}

Pixel*
Gis::getRandomPlotByValue(string layer_h, int layerValue_h){

  vector <Pixel* > candidates;
  vector <uint> counts;
  for(uint i=0;i<pxVector.size();i++) counts.push_back(i);
  random_shuffle(counts.begin(), counts.end()); // randomize the elements of the array.

  for (uint i=0;i<counts.size();i++){
    if(pxVector.at(counts.at(i)).getDoubleValue(layer_h) == layerValue_h ) {
        return &pxVector.at(counts.at(i));
    }
  }

  msgOut(MSG_CRITICAL_ERROR,"We can't find any plot with "+d2s(layerValue_h)+" value on layer "+layer_h+".");
  Pixel* toReturn;
  toReturn =0;
  return toReturn;
}
/**

@param layer_h      Name of the layer
@param layerValue_h Value we want the plots for
@param onlyFreePlots Flag to get only plots marked as free (d=false)
@param outputLevel Level of output in case of failure (no plots available). Default is warning, but if set as MSG_CRITICAL_ERROR it make stop the model.


*/
vector <Pixel*>
Gis::getAllPlotsByValue(string layer_h, int layerValue_h, int outputLevel){
  // this would be easier to mantain and cleaned code, but slighly slower:
  //vector<int> layerValues;
  //layerValues.push_back(layerValue_h);
  //return getAllPlotsByValue(layer_h, layerValues, onlyFreePlots, outputLevel);

  vector <Pixel* > candidates;
  for (uint i=0;i<pxVector.size();i++){
    if(pxVector.at(i).getDoubleValue(layer_h) == layerValue_h){
      candidates.push_back(&pxVector.at(i));
    }
  }

  if (candidates.size()>0){
    random_shuffle(candidates.begin(), candidates.end()); // randomize ther elements of the array... cool !!! ;-)))
  }
  else {
    msgOut(outputLevel,"We can't find any free plot with "+d2s(layerValue_h)+" value on layer "+layer_h+".");
  }
  return candidates;
}

/**
```

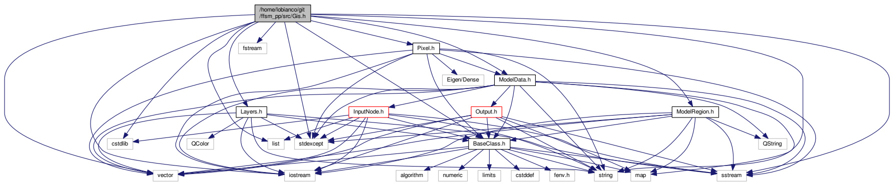
```

@param layer_h      Name of the layer
@param layerValues_h Values we want the plots for
@param onlyFreePlots Flag to get only plots marked as free (d=false)
@param outputLevel Level of output in case of failure (no plots available). Default is warning, but if set as MSG_CRITICAL_ERROR it make stop the model.


*/
vector <Pixel*>
Gis::getAllPlotsByValue(string layer_h, vector<int> layerValues_h, int outputLevel){
  vector <Pixel* > candidates;
  string valuesToMatch;
  unsigned int z;

  //string of the required land values to match;
  for (uint j=0;j<layerValues_h.size();j++){
    valuesToMatch = valuesToMatch + " " + i2s(layerValues_h.at(j));
  }

  for (uint i=0;i<pxVector.size();i++){
    z = valuesToMatch.find(d2s(pxVector.at(i).getDoubleValue(layer_h))); // search if in the string of required values is included also the value of the current plot
    if(z!=string::npos){ //z is not at the end of the string, means found!
      candidates.push_back(&pxVector.at(i));
    }
  }

  if (candidates.size()>0){
    random_shuffle(candidates.begin(), candidates.end()); // randomize ther elements of the array... cool !!! ;-)))
  }
  else {
    msgOut(outputLevel,"We can’t find any free plot with the specified values ("+valuesToMatch+") on layer "+layer_h+".");
  }
  return candidates;
}

/**

@param onlyFreePlots Flag to get only plots marked as free (d=false)
@param outputLevel Level of output in case of failure (no plots available). Default is warning, but if set as MSG_CRITICAL_ERROR it make stop the model.

*/
vector <Pixel*>
Gis::getAllPlots(int outputLevel){
  vector <Pixel* > candidates;
  for (uint i=0;i<pxVector.size();i++){
    candidates.push_back(&pxVector.at(i));
  }
  if (candidates.size()>0){
    random_shuffle(candidates.begin(), candidates.end()); // randomize ther elements of the array... cool !!! ;-)))
  }
  else {
    msgOut(outputLevel,"We can’t find any free plot.");
  }
  return candidates;
}

/// Return the vector of all plots by a specific region (main region or subregion), optionally shuffled;
vector <Pixel*>
Gis::getAllPlotsByRegion(ModelRegion &region_h, bool shuffle){
  vector <Pixel*> regionalPixels = region_h.getMyPixels();
  if(shuffle){
    random_shuffle(regionalPixels.begin(), regionalPixels.end()); // randomize the elements of the array.
  }
  return regionalPixels;
}

vector <Pixel*>
Gis::getAllPlotsByRegion(int regId_h, bool shuffle){
    ModelRegion* reg = MTHREAD->MD->getRegion(regId_h);
    return getAllPlotsByRegion(*reg,shuffle);
}



vector <string>
Gis::getLayerNames(){
  vector <string> toReturn;
  for (uint i=0;i<layerVector.size();i++){
    toReturn.push_back(layerVector[i].getName());
  }
  return toReturn;
```

```
}

vector <Layers*>
Gis::getLayerPointers(){
  vector <Layers*> toReturn;
  for (uint i=0;i<layerVector.size();i++){
    toReturn.push_back(&layerVector[i]);
  }
  return toReturn;
}

void
Gis::printDebugValues (string layerName_h, int min_h, int max_h){
  int min=min_h;
  int max;
  int ID, X, Y;
  string out;
  double value;
  //double noValue = MTHREAD->MD->getDoubleSetting("noValue");
  if (max_h==0){
    max= pxVector.size();
  }
  else {
    max = max_h;
  }
  msgOut(MSG_DEBUG,"Printing debug information for layer "+layerName_h+".");
  for (int i=min;i<max;i++){
    value = pxVector.at(i).getDoubleValue(layerName_h);
    if (value != noValue){
      ID    = i;
      X     = pxVector.at(i).getX();
      Y     = pxVector.at(i).getY();
      out = "Px. "+i2s(ID)+" ("+i2s(X)+","+i2s(Y)+"): "+d2s(value);
      msgOut(MSG_DEBUG,out);
    }
  }
}

/**
This function filter the region, placing noValue on the selected informative layer on pixels that are outside the subregion.
<br>It was thinked for speedup the development without have to run the whole model for testing each new implementation, but it can used to see what happen in the model when only a subset of the region is analysed.
*/
void
Gis::filterSubRegion(string layerName_h){
 subXL = 0;
 subYT = 0;
 subXR = xNPixels-1;
 subYB = yNPixels-1;
}

double
Gis::getDistance(const Pixel* px1, const Pixel* px2){
  return sqrt (
        pow ( (((double)px1->getX()) - ((double)px2->getX()))*xMetersByPixel,2)
        +
        pow ( (((double)px1->getY()) - ((double)px2->getY()))*yMetersByPixel,2)
      );
}



void
Gis::printLayers(string layerName_h){
  msgOut(MSG_DEBUG,"Printing the layers");
  int iteration = MTHREAD->SCD->getIteration(); // are we on the first year of the simulation ??
  if(layerName_h == ""){
    for (uint i=0;i<layerVector.size();i++){
      // not printing if we are in a not-0 iteration and the content of the map doesn't change
      if (!iteration || layerVector[i].getDynamicContent()) layerVector[i].print();
    }
  } else {
    for (uint i=0;i<layerVector.size();i++){
      if(layerVector[i].getName() == layerName_h){
        if (!iteration || layerVector[i].getDynamicContent()) layerVector[i].print();
        return;
      }
    }
    msgOut(MSG_ERROR, "Layer "+layerName_h+" unknow. No layer printed.");
  }
}

void
Gis::printBinMaps(string layerName_h){
  msgOut(MSG_DEBUG,"Printing the maps as images");
  int iteration = MTHREAD->SCD->getIteration(); // are we on the first year of the simulation ??
  if(layerName_h == ""){
    for (uint i=0;i<layerVector.size();i++){
      if (!iteration || layerVector[i].getDynamicContent()) {layerVector[i].printBinMap();}
    }
  } else {
    for (uint i=0;i<layerVector.size();i++){
      if(layerVector[i].getName() == layerName_h){
        if (!iteration || layerVector[i].getDynamicContent()) {layerVector[i].printBinMap();}
        return;
      }
    }
    msgOut(MSG_ERROR, "Layer "+layerName_h+" unknow. No layer printed.");
  }
}

int
Gis::sub2realID(int id_h){
  // IMPORTANT: this function is called at refreshGUI() times, so if there are output messages, call them with the option to NOT refresh the gui, otherwise we go to an infinite loop...
  return id_h;
}

void
Gis::unpack(const string& key, string& parName, string& forName, string& dClass, int& year) const{
  int parNameDelimiter = key.find("#",0);
  int forNameDelimiter = key.find("#",parNameDelimiter+1);
  int dClassDelimiter = key.find("#",forNameDelimiter+1);
  int yearDelimiter = key.find("#",dClassDelimiter+1);
  if (yearDelimiter == string::npos){
    msgOut(MSG_CRITICAL_ERROR, "Error in unpacking the key for the layer.");
  }
  parName.assign(key,0,parNameDelimiter);
  forName.assign(key,parNameDelimiter+1,forNameDelimiter-parNameDelimiter-1);
  dClass.assign(key,forNameDelimiter+1,dClassDelimiter-forNameDelimiter-1);
  string yearString="";
  yearString.assign(key,dClassDelimiter+1,yearDelimiter-dClassDelimiter-1);
  year = s2i(yearString);
}

void
Gis::swap(const int& swap_what){

  for(uint i=0;i<pxVector.size();i++) {
    pxVector[i].swap(swap_what);
  }

}
```

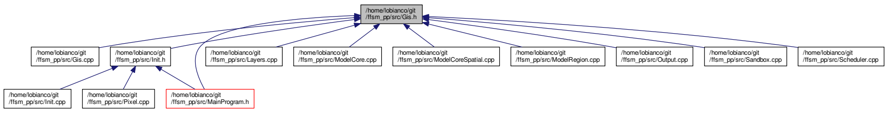
5.57 /home/lobianco/git/ffsm_pp/src/Gis.h File Reference

```
#include <cstdlib>
#include <list>
#include <string>
#include <vector>
#include <stdexcept>
#include <fstream>
#include <iostream>
#include <sstream>
#include "BaseClass.h"
#include "ModelData.h"
#include "Layers.h"
#include "Pixel.h"
#include "ModelRegion.h"
```

Include dependency graph for Gis.h:



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



Classes

- class Gis

  Class to manage the spatial dimension.

5.58 Gis.h

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d'Economie Forestière              *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify *
 *   it under the terms of the GNU General Public License as published by *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#ifndef STDGIS_H
#define STDGIS_H

#include <cstdlib>
#include <list>
#include <string>
#include <vector>
#include <stdexcept>
#include <fstream>
#include <iostream>
#include <sstream>

// regmas headers..
#include "BaseClass.h"
#include "ModelData.h"
#include "Layers.h"
#include "Pixel.h"
#include "ModelRegion.h"

using namespace std;
```

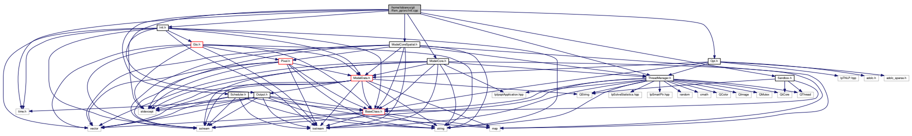
```

struct lUseCats;
struct reclassRules;
class Pixel;
class Agent_space;
class QImage;


/// Class to manage the spatial dimension
/**
Gis class is responsable to provide all methods for spatial analysis.
<br>It is equipped with two important vectors:
  - pxVector   contains the array of all pixels on the screen
  - layerVector contains the layer objects
<br>Along the model, IDs of pixels are assigned from left to right, from top to down:
<br>  --->
<br>   /
<br>  --->
<br>   /
<br>  --->
<p>Pixel origin (0,0) on the top left corner is also the system used by the underlying libraries, but put attencion that instead geographical coordinates, if we are on the North emisfere, are increasing along the up-right direction.

@author Antonello Lobianco
*/

class Gis: public BaseClass{

public:
                      Gis(ThreadManager* MTHREAD_h); ///< Constructor
                     ~Gis();
  /// Set the initial space environment, including loading data from files
  void                setSpace();
  /// Init the layers
  void                initLayers();
  void                initLayersPixelData();
  void                initLayersModelData(const int& year_h=DATA_NOW);
  /// Apply the forest reclassification with the rules defined in reclRules sheet
  void                applyForestReclassification();
  /// If subregion mode is on, this function place noValues on the selected layer for all out-of-region pixels
  void                filterSubRegion(string layerName_h);
  ///< Update the image behind a layer to the GUI;
  void                updateImage(string layerName_h);
  ///< Add one layer to the system
  void                addLayer(string name_h, string label_h, bool isInteger_h, bool dynamicContent_h, string fullFileName_h = "", bool display_h=true);
  ///< Fill a layer with empty values
  void                resetLayer(string layerName_h);
  ///< Check if a layer with a certain name is loaded in the model. Used e.g. to check if the dtm layer (optional) exist.
  bool                layerExist(const string & layerName_h, bool exactMatch = true) const;
  ///< Return a pointer to a layer given its name
  Layers*              getLayer(const string& layerName_h);
  ///< Add a legend item to an existing layer
  void                addLegendItem (
                          string       name_h,
                          int             D_h,
                          string      label_h,
                          int        rColor_h,
                          int        gColor_h,
                          int        bColor_h,
                          double   minValue_h,
                          double   maxValue_h   );
  /// Count the pixels within each legend item for the selected layer
  void                countItems(const string& layerName_h, const bool& debug=false);
  /// Return a pointer to a plot with a specific value for the specified layer
  Pixel*              getRandomPlotByValue(string layer_h, int layerValue__h);
  /// Return the vector (shuffled) of all plots with a specific value for a specified layer. It is also possible to specify the level in case of failure
  vector <Pixel*>     getAllPlotsByValue(string layer_h, int layerValue_h, int outputLevel=MSG_WARNING);
  /// Return the vector (shuffled) of all plots with specific values for a specified layer. It is also possible to specify the level in case of failure
  vector <Pixel*>     getAllPlotsByValue(string layer_h, vector<int> layerValues_h, int outputLevel=MSG_WARNING);
  /// Return the vector (shuffled) of all plots. It is also possible to specify the level in case of failure
  vector <Pixel*>     getAllPlots(int outputLevel=MSG_WARNING);
  /// Return the vector of all plots by a specific region (main region or subregion), optionally shuffled;
  vector <Pixel*>     getAllPlotsByRegion(ModelRegion &region_h, bool shuffle=false);
  vector <Pixel*>     getAllPlotsByRegion(int regId_h, bool shuffle=false);
  /// Return a vector of the layer ids (as string)
  vector <string>    getLayerNames();
  /// Return a vector of pointers of existing layers
  vector <Layers*>    getLayerPointers();
  /// Print the specified layer or all layers (if param layerName_h is missing). @see Layers::print()
  void                printLayers(string layerName_h="");
  /// Save an image in standard png format
  void                printBinMaps(string layerName_h="");


  ///< Print debug information (for each pixel in the requested interval, their values on the specified
 layer)
  void                printDebugValues (string layerName_h, int min_h=0, int max_h=0);
  double              getDistance(const Pixel* px1, const Pixel* px2);

  int                 getXNPixels() const {return xNPixels;};      ///< Return the number of
 pixels on X
  int                 getYNPixels() const {return yNPixels;};      ///< Return the number of
 pixels on Y
  double              getXyNPixels()const {return xyNPixels;};     ///< Return the total number
 of pixels
  double              getHaByPixel() const {return ((xMetersByPixel*yMetersByPixel)/10000) ;};
  double              getNoValue() const {return noValue;};
  Pixel*              getPixel(int x_h, int y_h){return &pxVector.at(x_h+y_h*xNPixels);};
///< Return a pixel pointer from its coordinates
  Pixel*              getPixel(int ID_h){return &pxVector.at(ID_h);};                   ///<
 Return a pixel pointer from its ID
  double              getGeoTopY() const {return geoTopY;};
  double              getGeoBottomY() const {return geoBottomY;};
  double              getGeoLeftX() const {return geoLeftX;};
  double              getGeoRightX() const {return geoRightX;};
  double              getXMetersByPixel() const {return xMetersByPixel;};
  double              getYMetersByPixel() const {return yMetersByPixel;};
  int                 getSubXL() const {return subXL;};
  int                 getSubXR() const {return subXR;};
  int                 getSubYT() const {return subYT;};
  int                 getSubYB() const {return subYB;};
  /// Transform the ID of a pixel in subregion coordinates to the real (and model used) coordinates
  int                 sub2realID(int id_h);
  string  pack(const string& parName, const string& forName, const string& dClass, const int& year)
 const {return parName+"#"+forName+"#"+dClass+"#"+i2s(year)+"#";};
  void unpack(const string& key, string& parName, string& forName, string& dClass, int& year) const;

  void                swap(const int &swap_what);

private:
  void                loadLayersDataFromFile(); ///< Load the data of a layer its datafile
  void          applySpatialStochasticValues(); ///< Apply stochastic simulation, e.g. regional volume
 growth s.d. -> tp multipliers
  void          applyStochasticRiskAdversion(); ///< Give to each agend a stochastic risk adversion. For
 now Pixel = Agent
  void                      cachePixelValues(); ///< For computational reasons cache some values in
 constant layers directly as properties of the pixel object
  vector <Pixel>                      pxVector; ///< array of Pixel objects
  vector <Layers>                  layerVector; ///< array of Layer objects
  vector <double>                   lUseTotals; ///< totals, in ha, of area in the region for
 each type (cached values)
  int                                 xNPixels; ///< number of pixels along the X dimension
  int                                 yNPixels; ///< number of pixels along the Y dimension
  double                             xyNPixels; ///< total number of pixels
  double                        xMetersByPixel; ///< pixel dimension (meters), X
  double                        yMetersByPixel; ///< pixel dimension (meters), Y
  double                              geoLeftX; ///< geo-coordinates of the map left border
  double                               geoTopY; ///< geo-coordinates of the map upper border
  double                             geoRightX; ///< geo-coordinates of the map right border
  double                            geoBottomY; ///< geo-coordinates of the map bottom border
  double                               noValue; ///< value internally use as novalue (individual
 layer maps can have other values)
  int                                    subXL; ///< sub region left X
  int                                    subXR; ///< sub region right X
  int                                    subYT; ///< sub region top Y
  int                                    subYB; ///< sub region bottom Y



};

#endif
```

5.59 /home/lobianco/git/ffsm_pp/src/Init.cpp File Reference

```
#include <time.h>
```

```
#include "Init.h"
#include "Scheduler.h"
#include "ThreadManager.h"
#include "Output.h"
#include "ModelCore.h"
#include "ModelCoreSpatial.h"
#include "Opt.h"
#include "Sandbox.h"
```

Include dependency graph for Init.cpp:



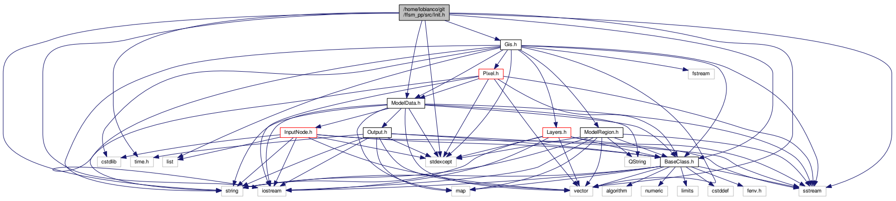
5.60 Init.cpp

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d'Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify *
 *   it under the terms of the GNU General Public License as published by *
 *   the Free Software Foundation; either version 3 of the License, or    *
 *   (at your option) any later version, given the compliance with the    *
 *   exceptions listed in the file COPYING that is distribued together    *
 *   with this file.                                                      *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,      *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of       *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the        *
 *   GNU General Public License for more details.                         *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License    *
 *   along with this program; if not, write to the                        *
 *   Free Software Foundation, Inc.,                                      *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.            *
 ***************************************************************************/
#include <time.h> // we only use this to seed the random number generator

#include "Init.h"
//#include "Pixel.h"
#include "Scheduler.h"
#include "ThreadManager.h"
#include "Output.h"
#include "ModelCore.h"
#include "ModelCoreSpatial.h"

#include "Opt.h"
#include "Sandbox.h"

//using namespace std;

Init::Init(ThreadManager* MTHREAD_h){
  MTHREAD=MTHREAD_h;
  InitState=0;
}

Init::~Init()
{
}

void
Init::setInitLevel(int level_h){

  switch (level_h){
    case 0:
      this->setInitLevel0();
      break;
    case 1:
      this->setInitLevel1();
      break;
    case 2:
      this->setInitLevel2();
      break;
    case 3:
      this->setInitLevel3();
      break;
    case 4:
      this->setInitLevel4();
      break;
    case 5:
      this->setInitLevel5();
      break;
    case 6:
      this->setInitLevel6();
      break;
    default:
      msgOut(MSG_ERROR,"unexpected Init level");
    }
}

void
Init::setInitLevel0(){
  //unused now
  InitState=0;
}

/**
Setting up the space
<br>Level 1 :
 - set the environment (settings, available resource name, possible activities)
 - init the space
@see ModelData::setDefaultSettings();
@see Gis::setSpace()
@see Manager_farmers::setAgentMoulds()

*/
void
Init::setInitLevel1(){
  //Loading data from file.
  InitState=1;
  msgOut(MSG_DEBUG,"Entering Init state "+i2s(InitState));
  time(&now);
  current = localtime(&now);
  string timemessage = "Local time is "+i2s(current->tm_hour)+":"+i2s(current->tm_min)+":"+ i2s(current->tm_sec);
  msgOut(MSG_INFO, timemessage);
  string scenarioName = MTHREAD->getScenarioName();
  MTHREAD->MD->setScenarioData(); // set the characteristics (including overriding tables of the scneario)
  MTHREAD->MD->setDefaultSettings();
  MTHREAD->MD->setScenarioSettings();
  if(MTHREAD->MD->getBoolSetting("newRandomSeed")){
    // See here for how to use the new C++11 random functions:
    // http://www.johndcook.com/cpp_TR1_random.html
    // usage example:
    // std::normal_distribution<double> d(100000,3);
    // double x = d(*MTHREAD->gen);
    srand(time(NULL));
    //std::random_device randev;
    //MTHREAD->gen = new std::mt19937(randev());
    MTHREAD->gen = new std::mt19937(time(0));

    //TO.DO change scenarioname to scenarioname_random number
    uniform_int_distribution<> ud(1, 1000000);
    int randomscenario = ud(*MTHREAD->gen);

    MTHREAD->setScenarioName(scenarioName+"_"+i2s(randomscenario));

  } else {
    MTHREAD->gen = new std::mt19937(NULL);
  }
  MTHREAD->SCD->setYear(MTHREAD->MD->getIntSetting("initialYear"));
  MTHREAD->MD->cacheSettings();

  MTHREAD->MD->createRegions();
  MTHREAD->MD->setDefaultForData();
  MTHREAD->MD->setScenarioForData();
  MTHREAD->MD->setDefaultProdData();
  MTHREAD->MD->setScenarioProdData();
  MTHREAD->MD->setForestTypes();
  MTHREAD->MD->setReclassificationRules();
  MTHREAD->MD->applyOverrides(); // Cancel all reg1 level data and trasform them in reg2 level if not already existing. Acts on forDataMap, prodDataMap and reclRules vectors
  MTHREAD->MD->setDefaultPathogenRules();
  MTHREAD->MD->setScenarioPathogenRules();
  MTHREAD->MD->setDefaultProductResourceMatrixLink();
  MTHREAD->MD->setScenarioProductResourceMatrixLink();
  MTHREAD->MD->applyDebugMode();
  MTHREAD->GIS->setSpace();
  MTHREAD->GIS->applyForestReclassification();
  MTHREAD->TEST->fullTest(); // normally empty function
}

void
Init::setInitLevel2(){
  InitState=2;
}

/**
Init 3 run the simulation/assign the values for the pre-optimisation year(s)
*/
void
Init::setInitLevel3(){
  InitState=3;
  MTHREAD->DO->initOutput();        // initialize the output files
  if(MTHREAD->MD->getBoolSetting("usePixelData")){
    MTHREAD->SCORE->runInitPeriod();
  } else {
    MTHREAD->CORE->runInitPeriod();
  }
}

void
Init::setInitLevel4(){
  InitState=4;
}

/**
Init level 5 pass the controll to the Scheduler object for the running of the simulations.
*/
void
Init::setInitLevel5(){
  InitState=5;
  MTHREAD->SCD->run(); // !!!! go "bello" !!!! start the simulation !!!!!
}

void
Init::setInitLevel6(){
  InitState=6;
  MTHREAD->DO->printFinalOutput();
  msgOut(MSG_INFO, "Model has ended scheduled simulation in a regular way.");
  time(&now);
  current = localtime(&now);
  string timemessage = "Local time is "+i2s(current->tm_hour)+":"+i2s(current->tm_min)+":"+ i2s(current->tm_sec);
  msgOut(MSG_INFO, timemessage);
}
```

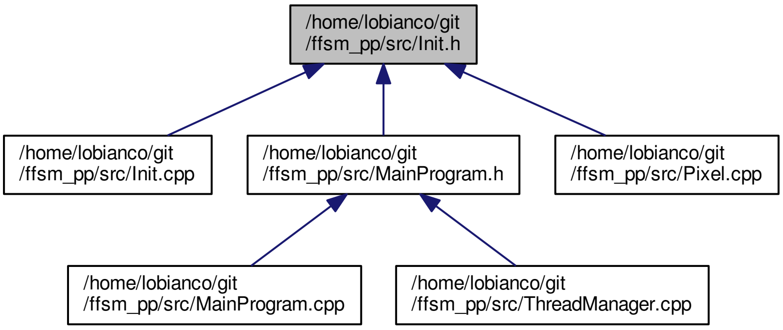
5.61 /home/lobianco/git/ffsm_pp/src/Init.h File Reference

```
#include <time.h>
#include <string>
#include <vector>
#include <stdexcept>
#include <iostream>
#include <sstream>
#include "BaseClass.h"
#include "ModelData.h"
#include "Gis.h"
```

Include dependency graph for Init.h:



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



Classes

- class Init

  Init the environment, the objects and the agents of the model

5.62 Init.h

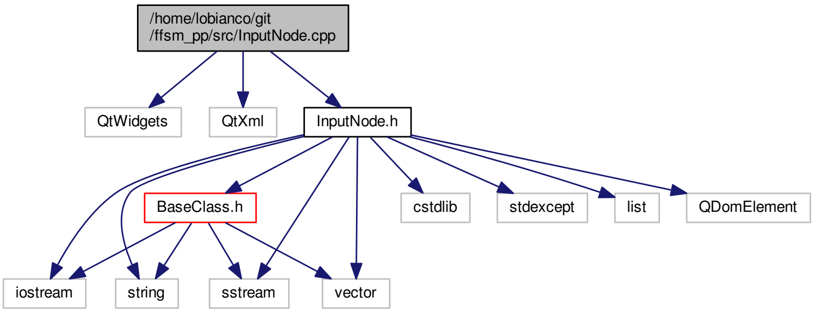
```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#ifndef STDINIT_H
#define STDINIT_H

#include <time.h>
```

```

#include <string>
#include <vector>
#include <stdexcept>
#include <iostream>
#include <sstream>

// FFSM include stuff..
#include "BaseClass.h"
#include "ModelData.h"
#include "Gis.h"

/// %Init the environment, the objects and the agents of the model
/**
The Init class is responsable to ask to the various objects to Init themself, in a 7-steps procedures.
<br>The basic idea is to first init the environment: options, settings and space.
<br>Then objects and agents are mould up, objects are assigned to agents and finally agents and objects are
 collocated in the space.
@author Antonello Lobianco
*/
class Init: public BaseClass{

public:

                      Init(ThreadManager* MTHREAD_h);
                     ~Init();
  /// Wrapper to the correct setInitLevelX()
  void                setInitLevel(int level_h);
  /// Unused, reserver for future use
  void                setInitLevel0();
  /// Setting up the space, the model objects and the agents (definitions only)
  void                setInitLevel1();
  /// Unused, reserver for future use
  void                setInitLevel2();
  /// Linking object to agents and assigning space proprieties to objects and agents
  void                setInitLevel3();
  /// Unused, reserver for future use
  void                setInitLevel4();
  /// Simulation start
  void                setInitLevel5();
  /// End of simulation (e.g. print summary statistics)
  void                setInitLevel6();
  int                 getInitState(){return InitState;};

private:
  int                                InitState; ///< One of the 7 possible init states (0..6)
  struct tm *current;
  time_t now;
};

#endif
```

5.63 /home/lobianco/git/ffsm_pp/src/InputNode.cpp File Reference

```
#include <QtWidgets>
#include <QtXml>
#include "InputNode.h"
```

Include dependency graph for InputNode.cpp:



5.64 InputNode.cpp

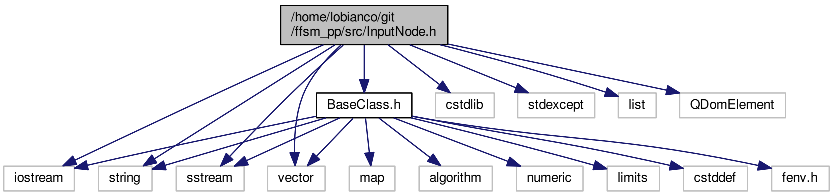
```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière                *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
//#include <QtGui>   // Qt4
#include <QtWidgets> // Qt5
#include <QtXml>

#include "InputNode.h"
//#include "InputDocument.h"


InputNode::InputNode(){
}

InputNode::~InputNode(){
}

bool
InputNode::setWorkingFile(std::string filename_h){

  QString errorStr;
  int errorLine;
  int errorColumn;

  QFile file(filename_h.c_str());
  QIODevice* device;
  device = &file;

  QDomDocument doc;
  if (!doc.setContent(device, true, &errorStr, &errorLine, &errorColumn)) {
    string message = "XML error on file "+ filename_h + " at line ";
    message.append(i2s(errorLine));
    message.append(" column ");
    message = message.c_str() + i2s(errorColumn);
    message = message + ": ";
    message = message + errorStr.toStdString();
    msgOut(MSG_WARNING, message.c_str());
    return false;
    }
  domElement = doc.documentElement();
  return true;
}

// *******************************************************************************
int
InputNode::getIntContent(){
  return domElement.text().toInt();
}

double
InputNode::getDoubleContent(){
  return domElement.text().toDouble(); // This is a Qt function that works both with dot and comma separators !
}

std::string
InputNode::getStringContent(){
  return domElement.text().toStdString();
}

bool
InputNode::getBoolContent(){
  string content = domElement.text().toStdString();
  if (content == "false" || content == "falso" || content == "FALSE" || content == "0")
    return false;
  else if (content == "true" || content == "vero" || content == "TRUE" || content == "1")
```

```
    return true;
  msgOut(MSG_WARNING, "Sorry, I don't know how to convert "+content+" to a bool value. I return true... hope for the best");
  return true;
}

int
InputNode::getIntAttributeByName(std::string attributeName_h){
  if (domElement.hasAttribute(attributeName_h.c_str())){
    return domElement.attribute(attributeName_h.c_str()).toInt();
  }else{
    msgOut(MSG_ERROR, "Element doens't have attribute " + attributeName_h );
    return 0;
  }
}

double
InputNode::getDoubleAttributeByName(std::string attributeName_h){
  if (domElement.hasAttribute(attributeName_h.c_str())){
    return domElement.attribute(attributeName_h.c_str()).toDouble();
  }else{
    msgOut(MSG_ERROR, "Element doens't have attribute " + attributeName_h );
    return 0;
  }
}

string
InputNode::getStringAttributeByName(std::string attributeName_h){
  if (domElement.hasAttribute(attributeName_h.c_str())){
    return domElement.attribute(attributeName_h.c_str()).toStdString();
  }else{
    msgOut(MSG_ERROR, "Element doens't have attribute " + attributeName_h );
    return "";
  }
}

bool
InputNode::hasAttributeByName(std::string attributeName_h){
  if (domElement.hasAttribute(attributeName_h.c_str())){
    return 1;
  }else{
    return 0;
  }
}

InputNode
InputNode::getNodeByName(string nodeName_h, int debugLevel, bool childFlag){
  /*
  QDomNodeList myElementList = domElement.elementsByTagName ( nodeName_h.c_str() );
  if (myElementList.size()>1){
    msgOut(debugLevel, "Too many elements. Expected only one of type "+nodeName_h);
  }
  if (myElementList.isEmpty()){
    msgOut(debugLevel, "No elements in the XML file. Expected 1 of type "+nodeName_h);
  }
  QDomElement myElement = myElementList.item(0).toElement() ;
  InputNode myInputNode(myElement);
  return myInputNode; */
  vector<InputNode> myNodes = getNodesByName(nodeName_h, debugLevel, childFlag);
  if (myNodes.size()>1){
    msgOut(debugLevel, "Too many elements. Expected only one of type "+nodeName_h);
    return myNodes[0];
  }
  if (myNodes.size() == 0){
    msgOut(debugLevel, "No elements in the XML file. Expected 1 of type "+nodeName_h+". Returning emty node!");
    InputNode toReturn;
    return toReturn;
  }
  return myNodes[0];
}

vector <InputNode>
InputNode::getNodesByName(string nodeName_h, int debugLevel, bool childFlag){
  vector <InputNode> myNodeVector;
  if (!childFlag){
    QDomNodeList myElementList = domElement.elementsByTagName ( nodeName_h.c_str() );
    for (int i=0;i<myElementList.size();i++){
      InputNode myInputNode(myElementList.item(i).toElement());
      myNodeVector.push_back(myInputNode);
    }

  }
  else {
    QDomNodeList myElementList = domElement.childNodes();
    for (int i=0;i<myElementList.size();i++){
      if (   myElementList.item(i).nodeType() == QDomNode::ElementNode
          && myElementList.item(i).toElement().tagName().toStdString() == nodeName_h){
        InputNode myInputNode(myElementList.item(i).toElement());
        myNodeVector.push_back(myInputNode);
      }
    }
  }
  if (myNodeVector.size()==0){
    msgOut(debugLevel, "No elements in the XML file. Expected at least one of type "+nodeName_h);
  }
  //for (int i=0;i<myElementList.size();i++){
  //  InputNode myInputNode(myElementList.item(i).toElement());
  //  myNodeVector.push_back(myInputNode);

    /*InputNode myInputNode(myElementList.item(i).toElement());
    string firstNodeContent= myInputNode.getStringContent();
    // checking that the setting is not an empy line nor a comment (e.g. starting with "#")..
    if(firstNodeContent=="") continue;
    unsigned int z;
    z = firstNodeContent.find("#");
    if( z!=string::npos && z == 0) continue;
    // chacking also the "childs" as in the XMLs deriving from csv I want delete the whole "<record>" tree, including his childs (fields)
    vector <InputNode> childs = myInputNode.getChildNodes();
    if(childs.size()>0){
      string firstChildContent= childs[0].getStringContent();
      // checking that the setting is not an empy line nor a comment (e.g. starting with "#")..
      if(firstChildContent=="") continue;
      unsigned int y;
      y = firstChildContent.find("#");
      if( y!=string::npos && y == 0) continue;
    }
    myNodeVector.push_back(myInputNode);
    */


  //}
  return myNodeVector;
}


/*
InputNode
InputNode::getNode(string nodeName_h, string attributeName_h, string attributeValue_h, int debugLevel){
  vector <InputNode> nodes = getNodes(nodeName_h, attributeName_h, attributeValue_h, debugLevel);
  if (nodes.size()>1){
    msgOut(debugLevel,"I got more than one node with specified carhacteristics. Returned the first one or aborting.");
    return nodes[0];
  } else if (nodes.size() == 0) {
    msgOut(debugLevel,"I don't have any node with the requested parameters. Returning an empty node.");
    InputNode toReturn;
    return toReturn;
  } else {
    return nodes[0];
  }
}

vector <InputNode>
InputNode::getNodes(string nodeName_h, string attributeName_h, string attributeValue_h, int debugLevel){
  vector <InputNode> nodes;

  return nodes;

}
*/


vector <InputNode>
InputNode::getChildNodes(){
  vector <InputNode> myNodeVector;
  QDomNodeList myElementList = domElement.childNodes();
  for (int i=0;i<myElementList.size();i++){
    if (myElementList.item(i).nodeType() == QDomNode::ElementNode ){
      InputNode myInputNode(myElementList.item(i).toElement());
      myNodeVector.push_back(myInputNode);
    }
  }
  return myNodeVector;
}

bool
InputNode::hasChildNode(string name_h){
  bool toReturn = false;
  QDomNodeList myElementList = domElement.childNodes();
  for (int i=0;i<myElementList.size();i++){
    if (myElementList.item(i).nodeType() == QDomNode::ElementNode ){
      if(myElementList.item(i).toElement().tagName().toStdString() == name_h) return true;
    }
  }
  return toReturn;
}

int
InputNode::getChildNodesCount(){
  int myElementListCountInt = 0;
  QDomNodeList myElementList = domElement.childNodes();
  for (int i=0;i<myElementList.size();i++){
    if (myElementList.item(i).nodeType() == QDomNode::ElementNode ){
      myElementListCountInt++ ;
    }
  }
  return myElementListCountInt;
}

string
InputNode::getNodeName(){
  return domElement.tagName().toStdString();
}

```

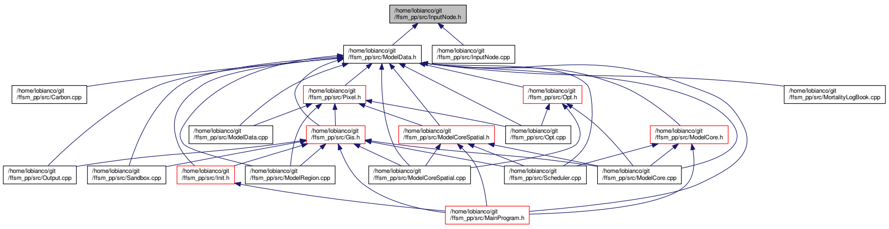
5.65 /home/lobianco/git/ffsm_pp/src/InputNode.h File Reference

```
#include <iostream>
#include <cstdlib>
#include <string>
#include <sstream>
#include <stdexcept>
#include <list>
#include <vector>
#include <QDomElement>
#include "BaseClass.h"
```

Include dependency graph for InputNode.h:



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



Classes

- class InputNode

  Wrapper around the underlying library for reading DOM elements (nodes).

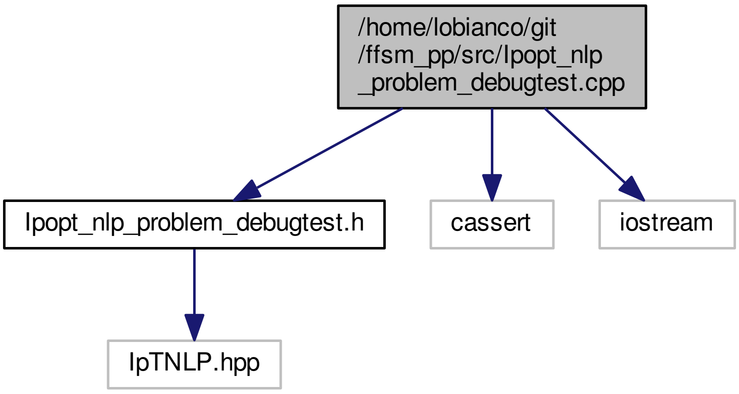
5.66 InputNode.h

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#ifndef INPUTNODE_H
#define INPUTNODE_H

#include <iostream>
#include <cstdlib>

#include <string>
#include <sstream>
#include <stdexcept>
#include <list>
#include <vector>

#include <QDomElement>

//regmas headers...
#include "BaseClass.h"

using namespace std;

//class QDomElement;

///Wrapper around the underlying library for reading DOM elements (nodes).

/**
A small wrapper class using an underlying library (currently QtXml) to read DOM nodes.
<br>This class works with the individual nodes (DOM Elements), while the companion class InputDocument wrapper the whole document (DOM Document).
<br>Note: In the DOM terminology "Elements" are a subset of the more general "nodes" (that include comments and other typologies..)
@author Antonello Lobianco
*/
class InputNode: public BaseClass{

public:
                      InputNode();
                      InputNode(QDomElement domElement_h){domElement=domElement_h;}; //<Constructor
                     ~InputNode();
  bool                setWorkingFile    (std::string  filename_h); ///< Load the file on memory. Return false if no success.
  int                 getIntContent();    ///< Get the content between its tagName as integer
  double              getDoubleContent(); ///< Get the content between its tagName as double
  string              getStringContent(); ///< Get the content between its tagName as std::string
  bool                getBoolContent();   ///< Get the content between its tagName as bool
  int                 getIntAttributeByName(string attributeName_h);    ///< Get an attribute by name as integer
  double              getDoubleAttributeByName(string attributeName_h); ///< Get an attribute by name as double
  string              getStringAttributeByName(string attributeName_h); ///< Get an attribute by name as string
  bool                hasAttributeByName(string attributeName_h);     ///< Check if an attribute with a certain name exist
  InputNode           getNodeByName     (string nodeName_h, int debugLevel=MSG_CRITICAL_ERROR, bool childFlag=false);  ///< return 0-or-1 nodes by name.
  vector <InputNode>  getNodesByName    (string nodeName_h, int debugLevel=MSG_WARNING, bool childFlag=false);  ///< return 0-to-n nodes by name
  /// Retrieve a child node with gived name and optionally with gived attribute or gived pair attribute/value. It raises an error if more than one.
  //InputNode           getNode(string nodeName_h, string attributeName_h="", string attributeValue_h="", int debugLevel=MSG_WARNING);
  /// Retrieve all child nodes with gived name and optionally with gived attribute or gived pair attribute/value. It raises an error if more than one.
  //vector <InputNode>          getNodes(string nodeName_h, string attributeName_h="", string
 attributeValue_h="", int debugLevel=MSG_WARNING);


  vector <InputNode>  getChildNodes();                            ///< Filtered to return only child
 <b>Elements</b>
  bool                hasChildNode(string name_h);                ///< True if it has specified child node
  int                 getChildNodesCount();                       ///< Only <b>Elements</b>
  string              getNodeName();

private:
  QDomElement                       domElement; ///< The underlying library-depending DOM
 rappresentation of the element

};

#endif
```

5.67 /home/lobianco/git/ffsm_pp/src/Ipopt_nlp_problem_debugtest.cpp File Reference

```
#include "Ipopt_nlp_problem_debugtest.h"
#include <cassert>
#include <iostream>
```

Include dependency graph for Ipopt_nlp_problem_debugtest.cpp:



5.68 Ipopt_nlp_problem_debugtest.cpp

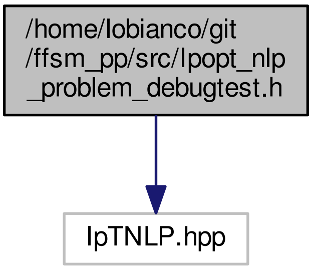
```
#include "Ipopt_nlp_problem_debugtest.h"

#include <cassert>
#include <iostream>

using namespace Ipopt;

// constructor
Ipopt_nlp_problem_debugtest::Ipopt_nlp_problem_debugtest
 ()
{}

//destructor
Ipopt_nlp_problem_debugtest::~Ipopt_nlp_problem_debugtest
 ()
{}

// returns the size of the problem
bool Ipopt_nlp_problem_debugtest::get_nlp_info(Index& n, Index& m,
 Index& nnz_jac_g,
                             Index& nnz_h_lag, IndexStyleEnum& index_style)
{
  // The problem described in Ipopt_nlp_problem_debugtest.hpp has 4 variables, x[0] through x[3]
  n = 4;

  // one equality constraint and one inequality constraint
  m = 2;

  // in this example the jacobian is dense and contains 8 nonzeros
  nnz_jac_g = 8;

  // the hessian is also dense and has 16 total nonzeros, but we
  // only need the lower left corner (since it is symmetric)
  nnz_h_lag = 10;

  // use the C style indexing (0-based)
  index_style = TNLP::C_STYLE;

  return true;
}

// returns the variable bounds
bool Ipopt_nlp_problem_debugtest::get_bounds_info(Index n, Number* x_l, Number* x_u,
                                Index m, Number* g_l, Number* g_u)
{
  // here, the n and m we gave IPOPT in get_nlp_info are passed back to us.
  // If desired, we could assert to make sure they are what we think they are.
  assert(n == 4);
  assert(m == 2);

  // the variables have lower bounds of 1
  for (Index i=0; i<4; i++) {
    x_l[i] = 1.0;
  }

  // the variables have upper bounds of 5
  for (Index i=0; i<4; i++) {
    x_u[i] = 5.0;
  }

  // the first constraint g1 has a lower bound of 25
  g_l[0] = 25;
  // the first constraint g1 has NO upper bound, here we set it to 2e19.
  // Ipopt interprets any number greater than nlp_upper_bound_inf as
  // infinity. The default value of nlp_upper_bound_inf and nlp_lower_bound_inf
  // is 1e19 and can be changed through ipopt options.
  g_u[0] = 2e19;

  // the second constraint g2 is an equality constraint, so we set the
  // upper and lower bound to the same value
  g_l[1] = g_u[1] = 40.0;

  return true;
}

// returns the initial point for the problem
bool Ipopt_nlp_problem_debugtest::get_starting_point(Index n, bool init_x, Number* x,
                                   bool init_z, Number* z_L, Number* z_U,
                                   Index m, bool init_lambda,
                                   Number* lambda)
{
  // Here, we assume we only have starting values for x, if you code
  // your own NLP, you can provide starting values for the dual variables
  // if you wish
  assert(init_x == true);
  assert(init_z == false);
  assert(init_lambda == false);

  // initialize to the given starting point
  x[0] = 1.0;
  x[1] = 5.0;
  x[2] = 5.0;
  x[3] = 1.0;

  return true;
}

// returns the value of the objective function
bool Ipopt_nlp_problem_debugtest::eval_f(Index n, const Number* x, bool new_x, Number& obj_value)
{
  assert(n == 4);

  obj_value = x[0] * x[3] * (x[0] + x[1] + x[2]) + x[2];

  return true;
}

// return the gradient of the objective function grad_{x} f(x)
bool Ipopt_nlp_problem_debugtest::eval_grad_f(Index n, const Number
* x, bool new_x, Number* grad_f)
{
  assert(n == 4);

  grad_f[0] = x[0] * x[3] + x[3] * (x[0] + x[1] + x[2]);
  grad_f[1] = x[0] * x[3];
  grad_f[2] = x[0] * x[3] + 1;
  grad_f[3] = x[0] * (x[0] + x[1] + x[2]);

  return true;
}

// return the value of the constraints: g(x)
bool Ipopt_nlp_problem_debugtest::eval_g(Index n, const Number* x, bool
new_x, Index m, Number* g)
{
  assert(n == 4);
  assert(m == 2);

  g[0] = x[0] * x[1] * x[2] * x[3];
  g[1] = x[0]*x[0] + x[1]*x[1] + x[2]*x[2] + x[3]*x[3];

  return true;
}

// return the structure or values of the jacobian
bool Ipopt_nlp_problem_debugtest::eval_jac_g(Index n, const Number*
x, bool new_x,
                           Index m, Index nele_jac, Index* iRow, Index *jCol,
                           Number* values)
{
  if (values == NULL) {
    // return the structure of the jacobian

    // this particular jacobian is dense
    iRow[0] = 0;
    jCol[0] = 0;
    iRow[1] = 0;
    jCol[1] = 1;
    iRow[2] = 0;
    jCol[2] = 2;
    iRow[3] = 0;
    jCol[3] = 3;
    iRow[4] = 1;
    jCol[4] = 0;
    iRow[5] = 1;
    jCol[5] = 1;
    iRow[6] = 1;
    jCol[6] = 2;
    iRow[7] = 1;
    jCol[7] = 3;
  }
  else {
    // return the values of the jacobian of the constraints

    values[0] = x[1]*x[2]*x[3]; // 0,0
    values[1] = x[0]*x[2]*x[3]; // 0,1
    values[2] = x[0]*x[1]*x[3]; // 0,2
    values[3] = x[0]*x[1]*x[2]; // 0,3

    values[4] = 2*x[0]; // 1,0
    values[5] = 2*x[1]; // 1,1
    values[6] = 2*x[2]; // 1,2
    values[7] = 2*x[3]; // 1,3
  }

  return true;
}


//return the structure or values of the hessian
bool Ipopt_nlp_problem_debugtest::eval_h(Index n, const Number* x, bool
new_x,
                       Number obj_factor, Index m, const Number* lambda,
                       bool new_lambda, Index nele_hess, Index* iRow,
                       Index* jCol, Number* values)
{
  if (values == NULL) {
    // return the structure. This is a symmetric matrix, fill the lower left
    // triangle only.

    // the hessian for this problem is actually dense
    Index idx=0;
    for (Index row = 0; row < 4; row++) {
      for (Index col = 0; col <= row; col++) {
        iRow[idx] = row;
        jCol[idx] = col;
        idx++;
      }
    }

    assert(idx == nele_hess);
  }
  else {
    // return the values. This is a symmetric matrix, fill the lower left
    // triangle only

    // fill the objective portion
    values[0] = obj_factor * (2*x[3]); // 0,0

    values[1] = obj_factor * (x[3]);   // 1,0
    values[2] = 0.;                    // 1,1

    values[3] = obj_factor * (x[3]);   // 2,0
    values[4] = 0.;                    // 2,1
    values[5] = 0.;                    // 2,2

    values[6] = obj_factor * (2*x[0] + x[1] + x[2]); // 3,0
    values[7] = obj_factor * (x[0]);                 // 3,1
    values[8] = obj_factor * (x[0]);                 // 3,2
    values[9] = 0.;                                  // 3,3


    // add the portion for the first constraint
    values[1] += lambda[0] * (x[2] * x[3]); // 1,0

    values[3] += lambda[0] * (x[1] * x[3]); // 2,0
    values[4] += lambda[0] * (x[0] * x[3]); // 2,1

    values[6] += lambda[0] * (x[1] * x[2]); // 3,0
    values[7] += lambda[0] * (x[0] * x[2]); // 3,1
    values[8] += lambda[0] * (x[0] * x[1]); // 3,2

    // add the portion for the second constraint
    values[0] += lambda[1] * 2; // 0,0

    values[2] += lambda[1] * 2; // 1,1

    values[5] += lambda[1] * 2; // 2,2

    values[9] += lambda[1] * 2; // 3,3
  }

  return true;
}



void Ipopt_nlp_problem_debugtest::finalize_solution(
SolverReturn status,
                                  Index n, const Number* x, const Number* z_L, const Number* z_U,
                                  Index m, const Number* g, const Number* lambda,
                                  Number obj_value,
                  const IpoptData* ip_data,
                  IpoptCalculatedQuantities* ip_cq)
{
  // here is where we would store the solution to variables, or write to a file, etc
  // so we could use the solution.

  // For this example, we write the solution to the console
  std::cout << std::endl << std::endl << "Solution of the primal variables, x" << std::endl;
  for (Index i=0; i<n; i++) {
     std::cout << "x[" << i << "] = " << x[i] << std::endl;
  }

  std::cout << std::endl << std::endl << "Solution of the bound multipliers, z_L and z_U" << std::endl;
  for (Index i=0; i<n; i++) {
    std::cout << "z_L[" << i << "] = " << z_L[i] << std::endl;
  }
  for (Index i=0; i<n; i++) {
    std::cout << "z_U[" << i << "] = " << z_U[i] << std::endl;
  }

  std::cout << std::endl << std::endl << "Objective value" << std::endl;
  std::cout << "f(x*) = " << obj_value << std::endl;

  std::cout << std::endl << "Final value of the constraints:" << std::endl;
  for (Index i=0; i<m ;i++) {
    std::cout << "g(" << i << ") = " << g[i] << std::endl;
```

```
00271   }
00272 }
```

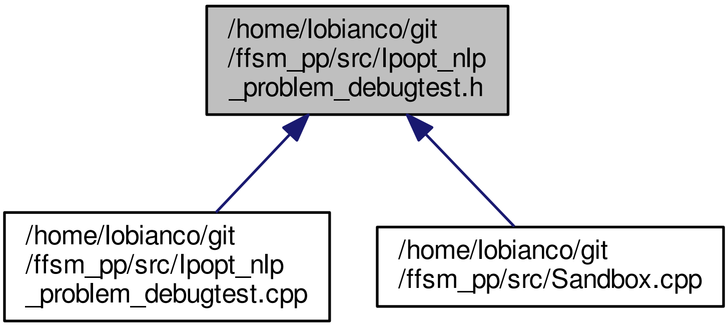
5.69 /home/lobianco/git/ffsm_pp/src/Ipopt_nlp_problem_debugtest.h File Reference

`#include "IpTNLP.hpp"`

Include dependency graph for Ipopt_nlp_problem_debugtest.h:



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



Classes

- class Ipopt_nlp_problem_debugtest

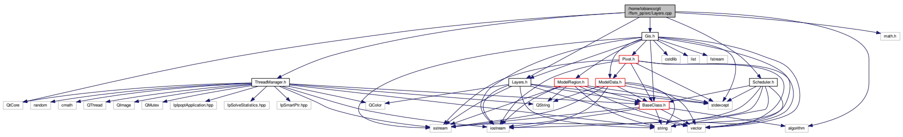
5.70 Ipopt_nlp_problem_debugtest.h

```
#ifndef IPOPT_NLP_PROBLEM_DEBUGTEST_H
#define IPOPT_NLP_PROBLEM_DEBUGTEST_H

#include "IpTNLP.hpp"

using namespace Ipopt;

/** C++ Example NLP for interfacing a problem with IPOPT.
 *  HS071_NLP implements a C++ example of problem 71 of the
 *  Hock-Schittkowski test suite. This example is designed to go
 *  along with the tutorial document and show how to interface
 *  with IPOPT through the TNLP interface.
 *
 * Problem hs071 looks like this
 *
 *     min   x1*x4*(x1 + x2 + x3)  +  x3
 *     s.t.  x1*x2*x3*x4                   >=  25
 *           x1**2 + x2**2 + x3**2 + x4**2  =  40
 *           1 <=  x1,x2,x3,x4  <= 5
 *
 *     Starting point:
 *        x = (1, 5, 5, 1)
 *
 *     Optimal solution:
 *        x = (1.00000000, 4.74299963, 3.82114998, 1.37940829)
 *
 *
 */
class Ipopt_nlp_problem_debugtest : public TNLP
{
public:
  /** default constructor */
  Ipopt_nlp_problem_debugtest();

  /** default destructor */
  virtual ~Ipopt_nlp_problem_debugtest();

  /**@name Overloaded from TNLP */
  //@{
  /** Method to return some info about the nlp */
  virtual bool get_nlp_info(Index& n, Index& m, Index& nnz_jac_g,
                            Index& nnz_h_lag, IndexStyleEnum& index_style);

  /** Method to return the bounds for my problem */
  virtual bool get_bounds_info(Index n, Number* x_l, Number* x_u,
                               Index m, Number* g_l, Number* g_u);

  /** Method to return the starting point for the algorithm */
  virtual bool get_starting_point(Index n, bool init_x, Number* x,
                                  bool init_z, Number* z_L, Number* z_U,
                                  Index m, bool init_lambda,
                                  Number* lambda);

  /** Method to return the objective value */
  virtual bool eval_f(Index n, const Number* x, bool new_x, Number& obj_value);

  /** Method to return the gradient of the objective */
  virtual bool eval_grad_f(Index n, const Number* x, bool new_x, Number* grad_f);

  /** Method to return the constraint residuals */
  virtual bool eval_g(Index n, const Number* x, bool new_x, Index m, Number* g);

  /** Method to return:
   *   1) The structure of the jacobian (if "values" is NULL)
   *   2) The values of the jacobian (if "values" is not NULL)
   */
  virtual bool eval_jac_g(Index n, const Number* x, bool new_x,
                          Index m, Index nele_jac, Index* iRow, Index *jCol,
                          Number* values);


  /** Method to return:
   *   1) The structure of the hessian of the lagrangian (if "values" is NULL)
   *   2) The values of the hessian of the lagrangian (if "values" is not NULL)
   */

  virtual bool eval_h(Index n, const Number* x, bool new_x,
                      Number obj_factor, Index m, const Number* lambda,
                      bool new_lambda, Index nele_hess, Index* iRow,
                      Index* jCol, Number* values);

  //@}

  /** @name Solution Methods */
  //@{
  /** This method is called when the algorithm is complete so the TNLP can store/write the solution */
  virtual void finalize_solution(SolverReturn status,
                                 Index n, const Number* x, const Number* z_L, const Number* z_U,
                                 Index m, const Number* g, const Number* lambda,
                                 Number obj_value,
                  const IpoptData* ip_data,
                  IpoptCalculatedQuantities* ip_cq);
  //@}

private:
  /**@name Methods to block default compiler methods.
   * The compiler automatically generates the following three methods.
   *  Since the default compiler implementation is generally not what
   *  you want (for all but the most simple classes), we usually
   *  put the declarations of these methods in the private section
   *  and never implement them. This prevents the compiler from
   *  implementing an incorrect "default" behavior without us
   *  knowing. (See Scott Meyers book, "Effective C++")
   *
   */
  //@{
  //  Ipopt_nlp_problem_debugtest();
  Ipopt_nlp_problem_debugtest (const
Ipopt_nlp_problem_debugtest&);
  Ipopt_nlp_problem_debugtest& operator=(const
Ipopt_nlp_problem_debugtest&);
  //@}
};


#endif // IPOPT_NLP_PROBLEM_H
```

5.71 /home/lobianco/git/ffsm_pp/src/Layers.cpp File Reference

```
#include <QtCore>
#include <math.h>
#include <algorithm>
#include "Layers.h"
#include "Gis.h"
#include "ThreadManager.h"
#include "Scheduler.h"
```

Include dependency graph for Layers.cpp:



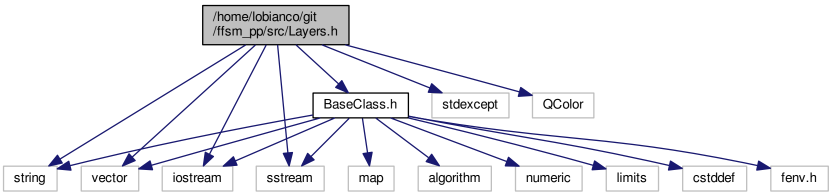
5.72 Layers.cpp

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#include <QtCore>

#include <math.h>
#include <algorithm>

#include "Layers.h"
#include "Gis.h"
#include "ThreadManager.h"
#include "Scheduler.h"

Layers::Layers(ThreadManager* MTHREAD_h, string name_h, string label_h, bool isInteger_h, bool dynamicContent_h, string fullFilename_h, bool display_h)
{
  MTHREAD=MTHREAD_h;
  name = name_h;
  label = label_h;
  isInteger = isInteger_h;
  dynamicContent = dynamicContent_h;
  fullFileName = fullFilename_h;
  display = display_h;
}

Layers::~Layers()
{
}

void
Layers::addLegendItem(int ID_h, string label_h, int rColor_h, int gColor_h, int bColor_h, double minValue_h, double maxValue_h){

  for (uint i=0;i<legendItems.size();i++){
    if (legendItems.at(i).ID == ID_h){
      msgOut(MSG_ERROR, "Trying to add a legend item that already exist on this layer (layer: "+label+" - legend label: "+label_h+")");
      //cout << "ID: "<<ID_h<<" Label: "<<label_h<<" minValue: "<<minValue_h << " maxValue: "<<maxValue_h<<endl;
      return;
    }
  }

  LegendItems ITEM;
  ITEM.ID = ID_h;
  ITEM.label = label_h;
  ITEM.rColor = rColor_h;
  ITEM.gColor = gColor_h;
  ITEM.bColor = bColor_h;
  ITEM.minValue = minValue_h;
  ITEM.maxValue = maxValue_h;
  ITEM.cashedCount=0;
  legendItems.push_back(ITEM);

}

void
Layers::addLegendItems(vector<LegendItems> legendItems_h){
  vector <LegendItems> toAdd;
  for(uint i=0; i<legendItems_h.size();i++){
    bool existing = false;
    for (uint j=0;j<legendItems.size();j++){
      if(legendItems_h[i].ID == legendItems[j].ID){
        existing = true;
        break;
      }
    }
    if(existing){
      msgOut(MSG_WARNING, "Legend item "+i2s(legendItems_h[i].ID)+" non added on layer "+this->name+" as already existing.");
    } else {
      toAdd.push_back(legendItems_h[i]);
    }
  }
  legendItems.insert( legendItems.end(), toAdd.begin(), toAdd.end() );
}


/**
Used in the init stage, this function take as input the real map code as just read from the map file, and filter it according to the reclassification rules.
@see ReclassRules
*/
double
Layers::filterExogenousDataset(double code_h){
  bool check =false;
  std::vector <double> cumPVector;
  std::vector <double> outCodesVector;
  double cumP = 0;
  double returnCode=0;

  for(uint i=0; i<reclassRulesVector.size(); i++){
    if (reclassRulesVector.at(i).inCode == code_h){
      check = true;
      cumP += reclassRulesVector.at(i).p;
      cumPVector.push_back(cumP);
      outCodesVector.push_back(reclassRulesVector.at(i).outCode);
    }
  }
  if (!check) {return code_h;}
  if (cumP <= 0.99999999 || cumP >= 1.00000001){msgOut(MSG_CRITICAL_ERROR,"the sum of land use reclassification rules is not 1 for at least one input code (input code:  "+d2s(code_h)+"; cumP: "+d2s(cumP)+")");}
  double random;
  //srand(time(NULL)); // this would re-initialise the random seed
  random = ((double)rand() / ((double)(RAND_MAX)+(double)(1)) );
  for(uint i=0; i<cumPVector.size(); i++){
    if (random <= cumPVector.at(i)){
      returnCode = outCodesVector.at(i);
      break;
    }
  }
  return returnCode;
}

/**
This function take as input the value stored in the pixel for the specific layer, loops over the legend item and find the one that match it, returning its color.
<br>If the layer is of type integer, the match is agains legendItem IDs, otherwise we compare the legendItem ranges.
@see LegendItems
*/
QColor
Layers::getColor(double ID_h){
  QColor nocolor(255,255,255);
  if (ID_h == MTHREAD->GIS->getNoValue()){
    return nocolor;
  }
  if (isInteger){
    for(uint i=0; i<legendItems.size(); i++){
      if (legendItems.at(i).ID == ((int)ID_h)){
        QColor color(legendItems.at(i).rColor, legendItems.at(i).gColor, legendItems.at(i).bColor);
        return color;
      }
    }
    return nocolor;
  }
  else {
    for(uint i=0; i<legendItems.size(); i++){
      if (ID_h < legendItems.at(i).maxValue &&  ID_h >= legendItems.at(i).minValue){
        QColor color(legendItems.at(i).rColor, legendItems.at(i).gColor, legendItems.at(i).bColor);
        return color;
      }
    }
    return nocolor;
  }
}
/**
This function take as input the value stored in the pixel for the specific layer, loops over the legend item and find the one that match it, returning its label.
<br>If the layer is of type integer, the match is agains legendItem IDs, otherwise we compare the legendItem ranges.
@see LegendItems
*/
string
Layers::getCategory(double ID_h){
  if (ID_h == MTHREAD->GIS->getNoValue()){
    return "";
  }
  if (isInteger){
    for(uint i=0; i<legendItems.size(); i++){
      if (legendItems.at(i).ID == ((int)ID_h)){
        return legendItems.at(i).label;
      }
    }
    return "";
  }
  else {
    for(uint i=0; i<legendItems.size(); i++){
      if (ID_h < legendItems.at(i).maxValue &&  ID_h >= legendItems.at(i).minValue){
        return legendItems.at(i).label;
      }
    }
    return "";
  }
}




void
Layers::countMyPixels(bool debug){

  for (uint i=0; i<legendItems.size(); i++){
    legendItems.at(i).cashedCount=0; //initialized with 0 values...
  }
  double totPixels = MTHREAD->GIS->getXyNPixels();
  double pixelValue;
  for (uint j=0;j<totPixels;j++){
    pixelValue = MTHREAD->GIS->getPixel(j)->getDoubleValue(name);
    if (isInteger){
      for(uint i=0; i<legendItems.size(); i++){
        if (legendItems.at(i).ID == ((int)pixelValue)){
          legendItems.at(i).cashedCount++;
          break;
        }
      }
    }
    else {
      for(uint i=0; i<legendItems.size(); i++){
        if (pixelValue < legendItems.at(i).maxValue &&  pixelValue >= legendItems.at(i).minValue){
          legendItems.at(i).cashedCount++;
          break;
        }
      }
    }
  }
  if (debug){
    msgOut(MSG_INFO, "Layer statistics - Count by Legend items");
    msgOut(MSG_INFO, "Layer name: "+label);
    msgOut(MSG_INFO, "Total plots: "+ d2s(totPixels));
    for(uint i=0;i<legendItems.size();i++){
      msgOut(MSG_INFO, legendItems.at(i).label+": "+i2s(legendItems.at(i).cashedCount));
    }
  }
}
void
Layers::randomShuffle(){


  vector <double> origValues;
  int maskValue = -MTHREAD->GIS->getNoValue();
  double totPixels = MTHREAD->GIS->getXyNPixels();
  for (uint i=0;i<totPixels;i++){
    double pxValue= MTHREAD->GIS->getPixel(i)->getDoubleValue(name);
    if(pxValue != MTHREAD->GIS->getNoValue()){
      origValues.push_back(pxValue);
      MTHREAD->GIS->getPixel(i)->changeValue(name,maskValue);
    }
  }
  random_shuffle(origValues.begin(), origValues.end()); // randomize the elements of the array.

  for (uint i=0;i<totPixels;i++){
    double pxValue= MTHREAD->GIS->getPixel(i)->getDoubleValue(name);
    if(pxValue != MTHREAD->GIS->getNoValue()){
      double toChangeValue = origValues.at(origValues.size()-1);
      //cout << toChangeValue << endl;
      origValues.pop_back();
      MTHREAD->GIS->getPixel(i)->changeValue(name,toChangeValue);
    }
  }

}
void
Layers::print(){

  if(MTHREAD->MD->getIntSetting("outputLevel")<OUTVL_MAPS) return;
  if(!display || !dynamicContent) return;
  string mapBaseDirectory = MTHREAD->MD->getBaseDirectory()+MTHREAD->MD->getOutputDirectory()+"maps/";
  string mapGridOutputDirectory = mapBaseDirectory+"asciiGrids/";
  string catsOutputDirectory = mapBaseDirectory+"cats/";
  string coloursOutputDirectory = mapBaseDirectory+"colr/";

  string mapFilename      = mapGridOutputDirectory +name+ "_" +i2s(MTHREAD->SCD->getYear()) +"_" +MTHREAD->getScenarioName();
  string catsFilename     = catsOutputDirectory    +name+ "_" +i2s(MTHREAD->SCD->getYear()) +"_" +MTHREAD->getScenarioName();
  string coloursFilename  = coloursOutputDirectory +name+ "_" +i2s(MTHREAD->SCD->getYear()) +"_" +MTHREAD->getScenarioName();
    string filenameListIntLayers = mapBaseDirectory+"integerListLayers/"+MTHREAD->getScenarioName();
    string filenameListFloatLayers = mapBaseDirectory+"floatListLayers/"+MTHREAD->getScenarioName();

  // printing the map...
  string header;
  if(MTHREAD->MD->getIntSetting("mapOutputFormat") == 1){ // GRASS ASCII Grid
    header =      "north: " + d2s(MTHREAD->GIS->getGeoTopY()) + "\n"
          + "south: " + d2s(MTHREAD->GIS->getGeoBottomY()) + "\n"
          + "east: " + d2s(MTHREAD->GIS->getGeoRightX()) + "\n"
          + "west: " + d2s(MTHREAD->GIS->getGeoLeftX()) + "\n"
          + "rows: " + i2s(MTHREAD->GIS->getYNPixels()) + "\n"
          + "cols: " + i2s(MTHREAD->GIS->getXNPixels()) + "\n"
          + "null: " + d2s(MTHREAD->GIS->getNoValue()) + "\n";

  } else if(MTHREAD->MD->getIntSetting("mapOutputFormat") == 2){
    header =      "ncols: " + i2s(MTHREAD->GIS->getXNPixels()) + "\n"
          + "nrows: " + i2s(MTHREAD->GIS->getYNPixels()) + "\n"
          + "xllcornel: " + d2s(MTHREAD->GIS->getGeoLeftX()) + "\n"
          + "yllcorner: " + d2s(MTHREAD->GIS->getGeoBottomY()) + "\n"
          + "cellsize: "  + d2s(MTHREAD->GIS->getXMetersByPixel()) + "\n"
          + "nodata_value: " + d2s(MTHREAD->GIS->getNoValue()) + "\n";
    if(MTHREAD->GIS->getXMetersByPixel() != MTHREAD->GIS->getYMetersByPixel()){
      msgOut(MSG_ERROR, "The X resolution is different to the Y resolution. I am exporting the map in ArcInfo ASCII Grid format using the X resolution, but be aware that it is incorrect, as this format doesn't support different X-Y resolutions.");
    }

  } else {
    msgOut(MSG_ERROR,"Map not print for unknow output type.");
  }

  ofstream outm; //out map
  outm.open(mapFilename.c_str(), ios::out); //ios::app to append..
  if (!outm){ msgOut(MSG_ERROR,"Error in opening the file "+mapFilename+".");}
  outm << header << "\n";

  for (int i=0;i<MTHREAD->GIS->getYNPixels();i++){
    for (int j=0;j<MTHREAD->GIS->getXNPixels();j++){
      outm << MTHREAD->GIS->getPixel(j, i)->getDoubleValue(name) << " ";
    }
    outm << "\n";
  }
  outm.close();

  //printing the cat file
  ofstream outc; //out category file
  outc.open(catsFilename.c_str(), ios::out); //ios::app to append..
  if (!outc){ msgOut(MSG_ERROR,"Error in opening the file "+catsFilename+".");}
  outc << "# " << name  << "_-_" << i2s(MTHREAD->SCD->getYear()) << "\n\n\n";
  outc << "0.00 0.00 0.00 0.00"<<"\n";

  if (isInteger){
    for(uint i=0;i<legendItems.size();i++){
      outc << legendItems[i].ID << ":"<< legendItems[i].label << "\n";
    }
  }
  else {
    for(uint i=0;i<legendItems.size();i++){
      outc << legendItems[i].minValue << ":"<< legendItems[i].maxValue << ":"<< legendItems[i].label << "\n";
    }
  }

  //printing the colour legend file
  ofstream outcl; //out colour file
  outcl.open(coloursFilename.c_str(), ios::out); //ios::app to append..
  if (!outcl){ msgOut(MSG_ERROR,"Error in opening the file "+coloursFilename+".");}
  outcl << "% " << name  << "_-_" << i2s(MTHREAD->SCD->getYear()) << "\n\n\n";

  if (isInteger){
    for(uint i=0;i<legendItems.size();i++){
      outcl << legendItems[i].ID << ":"<< legendItems[i].rColor << ":" << legendItems[i].gColor << ":" << legendItems[i].bColor << "\n";
    }
  }
  else {
    for(uint i=0;i<legendItems.size();i++){
      outcl << legendItems[i].minValue << ":"<< legendItems[i].rColor << ":" << legendItems[i].gColor << ":" << legendItems[i].bColor << " "<< legendItems[i].maxValue << ":"<< legendItems[i].rColor << ":" << legendItems[i].gColor << ":" << legendItems[i].bColor << "\n";
    }
  }

  // adding the layer to the list of saved layers..
  ofstream outList;
  if (isInteger){
    outList.open(filenameListIntLayers.c_str(), ios::app); // append !!!
    outList << name << "_" << MTHREAD->SCD->getYear() << "_" << MTHREAD->getScenarioName() << "\n";
  }
  else {
    outList.open(filenameListFloatLayers.c_str(), ios::app); // append !!!
    outList << name << "_" << MTHREAD->SCD->getYear() << "_" << MTHREAD->getScenarioName() << "\n";
  }
  outList.close();
}

void
Layers::printBinMap(){

  if(!display || !dynamicContent) return;

  int xNPixels             = MTHREAD->GIS->getXNPixels();
  int subXR                = MTHREAD->GIS->getSubXR();
  int subXL                = MTHREAD->GIS->getSubXL();
  int subYT                = MTHREAD->GIS->getSubYT();
  int subYB                = MTHREAD->GIS->getSubYB();

  string mapBaseDirectory = MTHREAD->MD->getBaseDirectory()+MTHREAD->MD->getOutputDirectory()+"maps/bitmaps/";
  string mapFilename      = mapBaseDirectory +name+ "_" +i2s(MTHREAD->SCD->getYear()) +"_" +MTHREAD->getScenarioName()+".png";

  QImage image = QImage(subXR-subXL+1, subYB-subYT+1, QImage::Format_RGB32);
  image.fill(qRgb(255, 255, 255));
  for (int countRow=subYT;countRow<subYB;countRow++){
    for (int countColumn=subXL;countColumn<subXR;countColumn++){
      double value = MTHREAD->GIS->getPixel(countRow*xNPixels+countColumn)->getDoubleValue(name);
      QColor color = this->getColor(value);
      image.setPixel(countColumn-subXL,countRow-subYT,color.rgb());
    }
  }
  image.save(mapFilename.c_str());
}
```

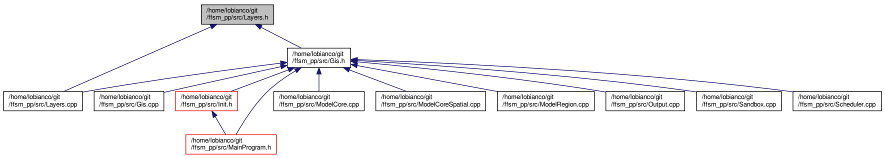
5.73 /home/lobianco/git/ffsm_pp/src/Layers.h File Reference

```
#include <string>
#include <vector>
#include <stdexcept>
#include <iostream>
#include <sstream>
#include <QColor>
#include "BaseClass.h"
```

Include dependency graph for Layers.h:



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


/home/lobianco/git /ffsm_pp/src/Layers.h
/home/lobianco/git /ffsm_pp/src/Gis.h
/home/lobianco/git /ffsm_pp/src/Layers.cpp
/home/lobianco/git /ffsm_pp/src/Gis.cpp
/home/lobianco/git /ffsm_pp/src/Init.h
/home/lobianco/git /ffsm_pp/src/ModelCore.cpp
/home/lobianco/git /ffsm_pp/src/ModelCoreSpatial.cpp
/home/lobianco/git /ffsm_pp/src/ModelRegion.cpp
/home/lobianco/git /ffsm_pp/src/Output.cpp
/home/lobianco/git /ffsm_pp/src/Sandbox.cpp
/home/lobianco/git /ffsm_pp/src/Scheduler.cpp
/home/lobianco/git /ffsm_pp/src/MainProgram.h

Classes

- class Layers

  Define layer objects at the regional level.
- struct LegendItems

  Legend items.
- struct ReclassRules

  Initial reclassification rules (dataset filters)

5.74 Layers.h

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#ifndef LAYERS_H
#define LAYERS_H
#include <string>
#include <vector>
#include <stdexcept>
#include <iostream>
#include <sstream>

#include <QColor>

// regmas headers...
#include "BaseClass.h"

using namespace std;

struct LegendItems;
struct ReclassRules;

/// Define layer objects at the regional level
/**
Layer class (setting, legend...)
<br>This class define layer objects, including:
 - a set of layer proprieties (name(ID), label, associated dataset, typology (integer or double)
 - a vector of legend items, associating one color to each value or interval
 - a vector of reclassification rule, when we need to work with a level of depth different of those coming with the dataset
@author Antonello Lobianco <antonello@regmas.org>
*/
class Layers : public BaseClass{
```

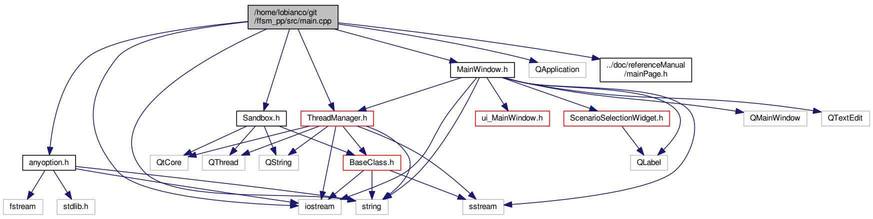
```

public:
  /// In the constructor we set the main layer properties
                      Layers(  ThreadManager*     MTHREAD_h,
                               string                 name_h,
                               string                label_h,
                               bool              isInteger_h,
                               bool         dynamicContent_h,
                               string         fullFilename_h,
                               bool               display_h=true);
                     ~Layers();
  /// Add a legend item. @see LegendItems
  void                addLegendItem( int              ID_h,
                                     string       label_h,
                                     int         rColor_h,
                                     int         gColor_h,
                                     int         bColor_h,
                                     double    minValue_h,
                                     double    maxValue_h   );
  void                addLegendItems(vector <LegendItems> legendItems_h);
  vector<LegendItems> getLegendItems(){return legendItems;};

  /// Evaluates all the legend items to find the one that match the input code, and return its color as a QColor
  QColor              getColor(double ID_h);
  /// Evaluates all the legend items to find the one that match the input code, and return its label
  string              getCategory(double ID_h);
  /// Used to reclassify the land use map for "generic" categories
  double              filterExogenousDataset(double code_h);
  /// Count the pixels going to each legend item and print them if debug==true
  void                countMyPixels(bool debug=false);
  /// For some sensitivity analisys, random the values for this layer for not-empty values (only integer layers)
  void                randomShuffle();
  /// Return if the layer is integer or not (If integer on each legend item: minValue==maxValue==ID)
  bool                getIsInteger() {return isInteger;}
  /// Print the layer content as an ASCII grid map with its companion files (classification and colors). It always print the whole region, even when subregion is actived.
  void                print();
  /// Print a binary reppresentation of the data (a standard image, e.g. a .png file). It prints only the subregion if this is active.
  void                printBinMap();

  string              getName() const {return name;}
  /// Return the filename of the associated dataset
  string              getFilename(){return fullFileName;}
  /// Return true if the content may change during simulation period
  bool                getDynamicContent(){return dynamicContent;}
  bool                getDisplay(){return display;}


private:
  string                                  name; ///< ID of the layer (no spaces allowed)
  string                                 label; ///< Label of the layer (spaces allowed)
  bool                               isInteger; ///< Type of the layer (true==integer, false==double. If true, on each legend item: minValue==maxValue==ID)
  bool                          dynamicContent; ///< True if the content may change during simulation year
  bool                                 display; ///< Normally true, but some layers used to just keep data shoudn't be normally processed
  string                          fullFileName; ///< Filename of the associated dataset (map)
  vector<LegendItems>              legendItems; ///< Vector of legend items. @see LegendItems
  vector<ReclassRules>      reclassRulesVector; ///< Vector of initial reclassification rules. @see ReclassRules
};

/// Legend items

/**
Struct containing data about the programm settings.
<br>The minValue and the maxValue are used to compare one record value and return the right color. If the layer is of type integer (isInteger==true), minValue==maxValue==ID.
@author Antonello Lobianco
*/
struct LegendItems {
  int                                       ID;
  string                                 label;
  int                                   rColor;
  int                                   gColor;
  int                                   bColor;
  double                              minValue;
  double                              maxValue;
  int                              cashedCount; ///< count the pixels whitin a item range
};

/// Initial reclassification rules (dataset filters)

```

```
00128 /**
00129 A structure for easy reclassification of "mixed" categories in some layers.
00130 <br>The reclassification can be made to both <i>increase</i> depth or <i>decrease</i> depth to the original
       dataset.
00131 <br>Eg, if in our model we don't differ between coniferous and hardwood forests, we can set all them to be
       "forest".
00132 <br>At the opposite, if our model require more detail than the map provide, e.g. irrigable arable VS dry
       arable, we can set the generic "arable land" of becoming "arable" or "dry" according with a regional-defined
       probability (getted from other sources, e.g. census data).
00133 @author Antonello Lobianco
00134 */
00135 struct ReclassRules{
00136   int                                     inCode;
00137   int                                    outCode;
00138   /// Probability that one pixel of code inCode will become of code outCode. 1 for fixed transformation.
00139   double                                       p;
00140 };
00141 
00142 #endif
```

5.75 /home/lobianco/git/ffsm_pp/src/main.cpp File Reference

```
#include <iostream>
#include <string>
#include "anyoption.h"
#include <QApplication>
#include "Sandbox.h"
#include "MainWindow.h"
#include "ThreadManager.h"
#include "../doc/referenceManual/mainPage.h"
```

Include dependency graph for main.cpp:



Functions

- int main (int argc, char ∗argv[ ])

5.75.1 Function Documentation

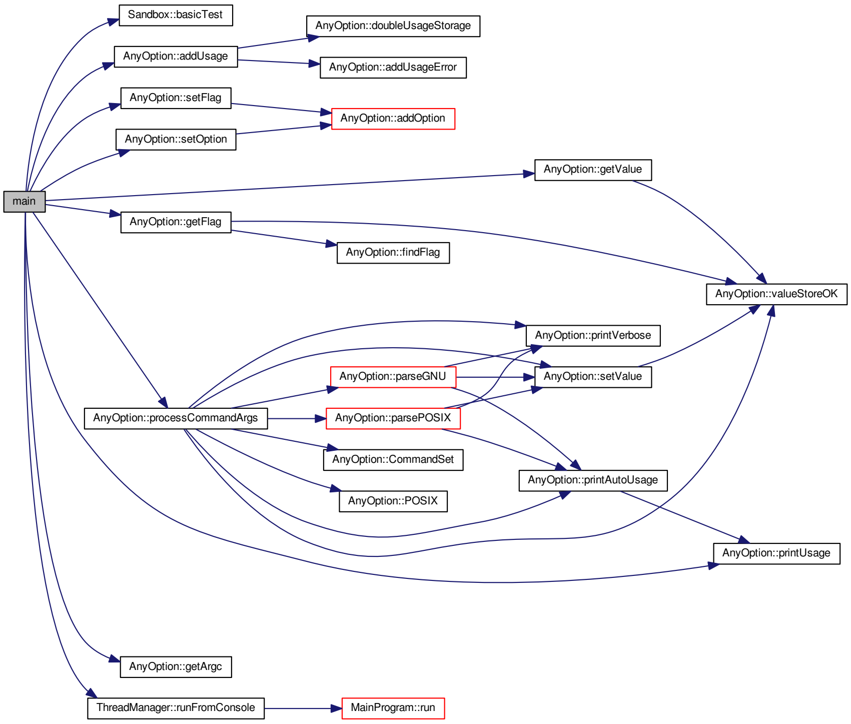
5.75.1.1 int main (int *argc*, char ∗ *argv[ ]*)

Definition at line 39 of file main.cpp.

```
00039                                   {
00040   #ifdef  __GNUC__
00041   #ifndef __MINGW32__
00042   // I can't use this automatic runtime error, as ADOL-C, for some reasons, has some places that
       explicitally create inf or nan
00043   //feenableexcept(FE_DIVBYZERO | FE_INVALID | FE_OVERFLOW); // to enable runtime error of division by zero
       (only in linux, not on MinGw)
00044   #endif
00045   #endif
```

```

  cout << endl;
  cout << "*******************************************************************" << endl;
  cout << "***    !! Welcome to FFSM - The Forest Sector Simulator !!    ***" << endl;
  cout << "***      For info & doc: http://www.ffsm-project.org/doc      ***" << endl;
  cout << "***         Compiled on: " << __DATE__ << " - " << __TIME__ << "                 ***" << endl;
  cout << "*******************************************************************" << endl<<endl;

  // Running "simple testing" that can be done at this early time
  Sandbox TEST;
  int debug=0;
  TEST.basicTest(); // normally this is an empty function, used only to place temporary in-developmentr tests
  //TEST.runSimpleTests();
  // TEST.testIpopt();
  //debug = TEST.testAdolc();
  //cout << "Early debug value: " << debug << endl;


  QDir dir;
  QString currentDir = dir.currentPath();
  // it's ok to leave the current directory (relative as where we are starting the application) rather than the application
  // path (relative to where ffsm is). This influence only the command line, where the -i option is always realtive to the local
  // position we are calling it from.


  QString inputFileName = "";
  QString scenarioName = "";


  // 1. CREATE AN OBJECT
  AnyOption *opt = new AnyOption();


  // 2. SET PREFERENCES
  //opt->setVerbose(); // print warnings about unknown options
  //opt->autoUsagePrint(true); // print usage for bad options

  // 3. SET THE USAGE/HELP
  opt->addUsage( "*** FFSM - Forest Sector Simulator ***" );
  opt->addUsage( "Usage: " );
  opt->addUsage( "" );
  opt->addUsage( " -h  --help      Prints this help " );
  opt->addUsage( " -c  --console       Run in console mode (no gui, default: false) " );
  opt->addUsage( " -i  --input_file [input_file_name]  Input file (relative path, default: 'data/ffsmInput.ods') " );
  opt->addUsage( " -s  --scenario [scenario_name]  Scenario name (default: the first defined in the input file) " );
  opt->addUsage( "" );
  opt->addUsage( "Notes:");
    opt->addUsage( "  - input_file and scenario options have no effect in GUI mode;" );
  opt->addUsage( "  - the working directory is the base path relative to the input file." );
  opt->addUsage( "" );
  opt->addUsage( "Read installed documentation or browse it at http://www.ffsm-project.org/doc." );
  opt->addUsage( "" );

  // 4. SET THE OPTION STRINGS/CHARACTERS
  opt->setFlag(  "help", 'h' );
  opt->setFlag(  "console", 'c' );
  opt->setOption(  "input_file", 'i' );
  opt->setOption(  "scenario", 's' );

  // 5. PROCESS THE COMMANDLINE
  opt->processCommandArgs( argc, argv );

  // 6. GET THE VALUES
  if( opt->getFlag( "help" ) || opt->getFlag( 'h' ) || opt->getArgc() >0 ) {
    opt->printUsage();
    delete opt;
    return EXIT_FAILURE;
  }

  if( opt->getValue( 'i' ) != NULL  || opt->getValue( "input_file" ) != NULL  ){
      QString tempdata(opt->getValue( 'i' ));
      inputFileName = currentDir + "/" + tempdata;
  }
  else {
      inputFileName = currentDir + "/data/ffsmInput.ods";
  }

  if( opt->getValue( 's' ) != NULL  || opt->getValue( "scenario" ) != NULL  ){
      scenarioName = opt->getValue( 's' );
  }

  if( opt->getFlag( ’c’ ) || opt->getFlag( "console" ) ){
        ThreadManager  modelMainThread;
        modelMainThread.runFromConsole(inputFileName,scenarioName);
  }
  else {
    QApplication app(argc, argv);
    MainWindow mainWin;
    mainWin.show();
    return app.exec();
  }
  delete opt;
}
```

Here is the call graph for this function:



5.76 main.cpp

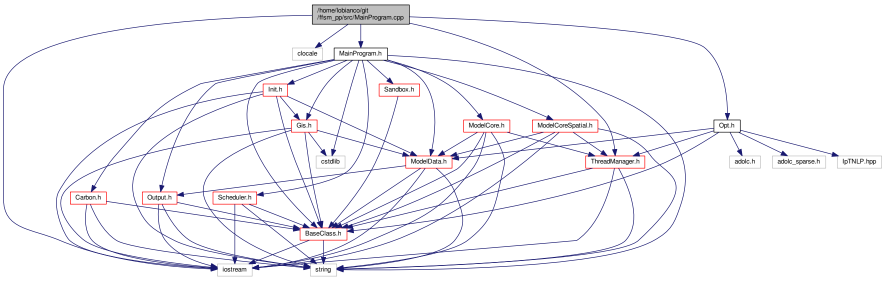
```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#include <iostream>
#include <string>
```

```

#include "anyoption.h"


#include <QApplication>

#include "Sandbox.h"
#include "MainWindow.h"
#include "ThreadManager.h"

// HTML code for the home page of the doxygen-generated documentation (Reference Manual)...
#include "../doc/referenceManual/mainPage.h"

using namespace std;

int main(int argc, char *argv[]){
  #ifdef  __GNUC__
  #ifndef __MINGW32__
  // I can't use this automatic runtime error, as ADOL-C, for some reasons, has some places that explicitly create inf or nan
  //feenableexcept(FE_DIVBYZERO | FE_INVALID | FE_OVERFLOW); // to enable runtime error of division by zero (only in linux, not on MinGw)
  #endif
  #endif

  cout << endl;
  cout << "*******************************************************************" << endl;
  cout << "***    !! Welcome to FFSM - The Forest Sector Simulator !!    ***" << endl;
  cout << "***      For info & doc: http://www.ffsm-project.org/doc      ***" << endl;
  cout << "***         Compiled on: " << __DATE__ << " - " << __TIME__ << "                 ***" << endl;
  cout << "*******************************************************************" << endl<<endl;

  // Running "simple testing" that can be done at this early time
  Sandbox TEST;
  int debug=0;
  TEST.basicTest(); // normally this is an empty function, used only to place temporary in-developmentr tests
  //TEST.runSimpleTests();
  // TEST.testIpopt();
  //debug = TEST.testAdolc();
  //cout << "Early debug value: " << debug << endl;


  QDir dir;
  QString currentDir = dir.currentPath();
  // it's ok to leave the current directory (relative as where we are starting the application) rather than the application
  // path (relative to where ffsm is). This influence only the command line, where the -i option is always realtive to the local
  // position we are calling it from.


  QString inputFileName = "";
  QString scenarioName = "";


  // 1. CREATE AN OBJECT
  AnyOption *opt = new AnyOption();


  // 2. SET PREFERENCES
  //opt->setVerbose(); // print warnings about unknown options
  //opt->autoUsagePrint(true); // print usage for bad options

  // 3. SET THE USAGE/HELP
  opt->addUsage( "*** FFSM - Forest Sector Simulator ***" );
  opt->addUsage( "Usage: " );
  opt->addUsage( "" );
  opt->addUsage( " -h  --help      Prints this help " );
  opt->addUsage( " -c  --console       Run in console mode (no gui, default: false) " );
  opt->addUsage( " -i  --input_file [input_file_name]  Input file (relative path, default: 'data/ffsmInput.ods') " );
  opt->addUsage( " -s  --scenario [scenario_name]  Scenario name (default: the first defined in the input file) " );
  opt->addUsage( "" );
  opt->addUsage( "Notes:");
    opt->addUsage( "  - input_file and scenario options have no effect in GUI mode;" );
  opt->addUsage( "  - the working directory is the base path relative to the input file." );
  opt->addUsage( "" );
  opt->addUsage( "Read installed documentation or browse it at http://www.ffsm-project.org/doc." );
  opt->addUsage( "" );

  // 4. SET THE OPTION STRINGS/CHARACTERS
  opt->setFlag(  "help", 'h' );
  opt->setFlag(  "console", 'c' );
  opt->setOption(  "input_file", 'i' );
  opt->setOption(  "scenario", 's' );

  // 5. PROCESS THE COMMANDLINE
  opt->processCommandArgs( argc, argv );

  // 6. GET THE VALUES
  if( opt->getFlag( "help" ) || opt->getFlag( 'h' ) || opt->
getArgc() >0 ) {
    opt->printUsage();
    delete opt;
    return EXIT_FAILURE;
  }

  if( opt->getValue( 'i' ) != NULL  || opt->getValue( "input_file" ) != NULL  ){
      QString tempdata(opt->getValue( 'i' ));
      inputFileName = currentDir + "/" + tempdata;
  }
  else {
      inputFileName = currentDir + "/data/ffsmInput.ods";
  }

  if( opt->getValue( 's' ) != NULL  || opt->getValue( "scenario" ) != NULL  ){
      scenarioName = opt->getValue( 's' );
  }

  if( opt->getFlag( 'c' ) || opt->getFlag( "console" ) ){
        ThreadManager  modelMainThread;
        modelMainThread.runFromConsole(inputFileName,scenarioName);
  }
  else {
    QApplication app(argc, argv);
    MainWindow mainWin;
    mainWin.show();
    return app.exec();
  }
  delete opt;
}
```

5.77 /home/lobianco/git/ffsm_pp/src/MainProgram.cpp File Reference

```
#include <iostream>
#include <clocale>
#include "MainProgram.h"
#include "ThreadManager.h"
#include "Opt.h"
```

Include dependency graph for MainProgram.cpp:



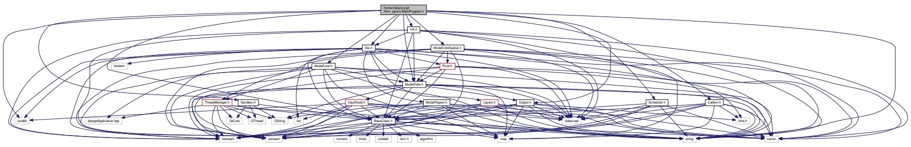
5.78 MainProgram.cpp

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                        *
 *                                                                          *
 *   This program is distributed in the hope that it will be useful,        *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of         *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the          *
 *   GNU General Public License for more details.                           *
 *                                                                          *
 *   You should have received a copy of the GNU General Public License      *
 *   along with this program; if not, write to the                          *
 *   Free Software Foundation, Inc.,                                        *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.              *
 ****************************************************************************/
#include <iostream>
#include <clocale>

#include "MainProgram.h"
#include "ThreadManager.h"
#include "Opt.h"




//constructor
MainProgram::MainProgram(ThreadManager* MTHREAD_h)
{
  //input_filename = input_filename_h;
  MTHREAD = MTHREAD_h;
  // Creating objects for the program flow:
  // the regional data object..
  ModelData *MD   = new ModelData(MTHREAD);
  MTHREAD->setMDPointer(MD);
  MTHREAD->MD->setBaseDiretory(MTHREAD->getBaseDirectory());
  MTHREAD->MD->loadInput(); // Unzip the ooffice input file and load it into memory

}

//distructor
MainProgram::~MainProgram(){

}

/**
This is the main call of the program.
<br>It firstly create the objects (and keep track of them trough pointers) of the main functional objects of the program.
<br>Then it call the INIT object to do its jobs and when it ends, it gives control to SCD (Scheduler) for the year loops.
<br>Finally it clean-up and returns.
*/
void
MainProgram::run(){

  setlocale(LC_ALL, "C"); // force to use the dot as digital separator also if we are running under the GUI

  // GIS information and methods..
  Gis *GIS  = new Gis(MTHREAD);
  MTHREAD->setGISPointer(GIS);
  // a test object for various 0-effects tests (sandbox)..
  Sandbox* TEST = new Sandbox(MTHREAD);
  MTHREAD->setTestPointer(TEST);
  // the Init object, it schedule the pre-simulation phase..
  Init *INIT = new Init(MTHREAD);
  MTHREAD->setINITPointer(INIT);
  // the scheduler object. It manage the simulation loops..
  Scheduler *SCD  = new Scheduler(MTHREAD);
  MTHREAD->setSCDPointer(SCD);
  // the core of the model
  ModelCore *CORE = new ModelCore(MTHREAD);
  MTHREAD->setCOREPointer(CORE);
  // the core of the model (spatial version)
  ModelCoreSpatial *SCORE = new ModelCoreSpatial(MTHREAD);
  MTHREAD->setSCOREPointer(SCORE);
  // the market optimisation algorithm
  Opt *OPT = new Opt(MTHREAD);
  MTHREAD->setOPTPointer(OPT);
  // manage the printing of data needed for scenario-analisys. The "message output" (needed to see "what is it happening?" are instead simply printed with msgOut()..
  Output *DO = new Output(MTHREAD);
  MTHREAD->setDOPointer(DO);
  // the carbon balance
  Carbon *CBAL = new Carbon(MTHREAD);
  MTHREAD->setCBALPointer(CBAL);

  // Creating an istance of INIT and delegating to it the Initialization phase..
  MTHREAD->INIT->setInitLevel(1); // Initial environment setting and agent rising
  refreshGUI();
  MTHREAD->INIT->setInitLevel(3); // assigning resources to agents and evenutal env reallocation
  refreshGUI();
  MTHREAD->INIT->setInitLevel(5); // starting simulations. Once INIT has ended it is the turn of SCD (Scheduler) to manage the simulation...
  refreshGUI();
  MTHREAD->INIT->setInitLevel(6); // ending simulations
  refreshGUI();

  // Deleting the pointers...
  // 20070102: if I delete the pointers I can not access the legend after simulation has ended
  // 20070109: pointers (e.g. INIT) are deleted in ThreadManager when a new simulation start
}

```

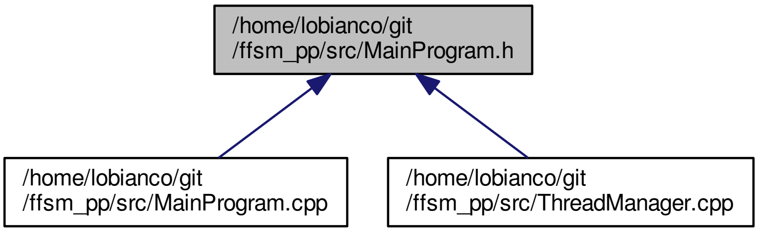
5.79 /home/lobianco/git/ffsm_pp/src/MainProgram.h File Reference

```
#include <cstdlib>
#include <string>
#include "BaseClass.h"
#include "ModelData.h"
#include "Gis.h"
#include "Init.h"
#include "Scheduler.h"
#include "Sandbox.h"
#include "Output.h"
#include "ModelCore.h"
#include "ModelCoreSpatial.h"
#include "Carbon.h"
```

Include dependency graph for MainProgram.h:



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


/home/lobianco/git
/ffsm_pp/src/MainProgram.h
/home/lobianco/git
/ffsm_pp/src/MainProgram.cpp
/home/lobianco/git
/ffsm_pp/src/ThreadManager.cpp

Classes

- class MainProgram

  Main program scheleton. It control the flow of the program.

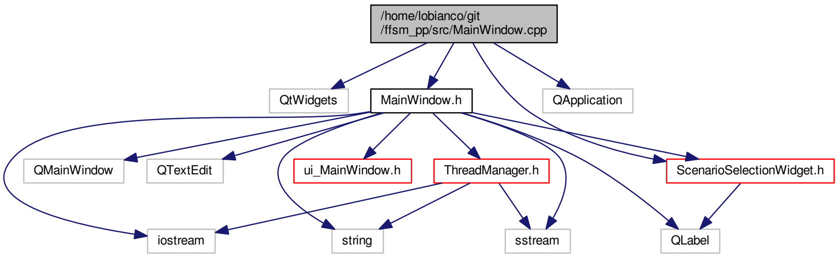
5.80 MainProgram.h

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d'Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#ifndef MAINPROGRAM_H
#define MAINPROGRAM_H

// standard include
#include <cstdlib>
#include <string>

// regmas headers...
#include "BaseClass.h"
#include "ModelData.h"
#include "Gis.h"
#include "Init.h"
#include "Scheduler.h"
#include "Sandbox.h"
#include "Output.h"
#include "ModelCore.h"
#include "ModelCoreSpatial.h"
#include "Carbon.h"

/// Main program scheleton. It control the flow of the program.

/**
There is only one istance of this class. It is responsable to load the setting files, call the Init class, "speack" with the Scheduler and finally end the program.
@author Antonello Lobianco
*/
class MainProgram: public BaseClass {

public:
                      MainProgram(ThreadManager* MTHREAD);
             ~MainProgram();
  void                run(); ///< Run the program

};

#endif
```

5.81 /home/lobianco/git/ffsm_pp/src/MainWindow.cpp File Reference

```
#include <QtWidgets>
#include "MainWindow.h"
#include "ScenarioSelectionWidget.h"
#include "QApplication"
```

Include dependency graph for MainWindow.cpp:



5.82 MainWindow.cpp

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distributed together    *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
//#include <QtGui>   // Qt4
#include <QtWidgets> // Qt5

#include "MainWindow.h"
#include "ScenarioSelectionWidget.h"
#include "QApplication"

using namespace std;


// *************** Initializzation functions...   *******************

/**

It setup the Gui from the QTDesiger autogenerated code and connect various GUI signal/slots

*/
MainWindow::MainWindow() {
  yearSBLabel=NULL;
  mainSBLabel=NULL;
  for (uint i=0;i<MaxRecentFiles;i++) recentFileActions[i] = NULL;
  separatorAction=NULL;

  setupUi(this);
  unsavedStatus=false;
  curModelFileName="data/ffsmInput.ods";
  curBaseDirectory = QApplication::applicationDirPath();
  curBaseDirectory.append("/data/");
  //curBaseDirectory = "data/";
  outputDirName="output/";
  setCurrentLogFileName("");
  createStatusBar();
  curLogFileName ="";
  debugMsgsEnable = true;

  for (int i = 0; i < MaxRecentFiles; ++i) {
    recentFileActions[i] = new QAction(this);
    recentFileActions[i]->setVisible(false);
    connect(recentFileActions[i], SIGNAL(triggered()), this, SLOT(openRecentFile()));
  }

  separatorAction = menuFile->addSeparator();
  for (int i = 0; i < MaxRecentFiles; ++i)
    menuFile->addAction(recentFileActions[i]);
  menuFile->addSeparator();
  menuFile->addAction(actionExit);

  readSettings();
  modelMainThread.setInputFileName(curModelFileName);
  //modelMainThread.setBaseDirectory(curBaseDirectory);

  // Status viewer....
  statusView->setColumnCount(2);
  statusView->setHeaderLabels(QStringList()<< tr ("Label") << tr ("Value"));
  statusView->clear();
  statusView->sortByColumn(0);
  statusView->setFocus(); //????




  /*
  DONE: statusView should be implemented like this:

  Model
    -> year
    -> total plots
    -> rented plots
    -> abandoned plots
  Managers
    -> Manager_farmer
      -> number of agents
  Agents
    Agent_0
      -> Type
      -> ID
      -> mould
      -> owned plots
      ...
    Agent_1
      -> Type
      -> ID
      -> mould
      -> owned plots
      ...
    ...
  */

  qRegisterMetaType<string>("string"); // allows string objects to be thread-safely queued within signal-slots comunications
  qRegisterMetaType<QString>("QString");
  qRegisterMetaType< QVector<QString> >("QVector<QString>");


  connect(actionRun, SIGNAL(triggered()), this, SLOT(startModelMainThread()));
  connect(actionPause, SIGNAL(triggered()), this, SLOT(pauseOrResumeModelMainThread()));
  connect(actionStop, SIGNAL(triggered()), this, SLOT(stopModelMainThread()));
  connect(actionExit, SIGNAL(triggered()), this, SLOT(close()));
  connect(actionSaveLog, SIGNAL(triggered()), this, SLOT(save()));
  connect(actionSaveLogAs, SIGNAL(triggered()), this, SLOT(saveAs()));
  connect(actionLoadConfiguration, SIGNAL(triggered()), this, SLOT(open()));
  connect(actionHideDebugMsgs, SIGNAL(triggered(bool)), this, SLOT(hideDebugMsgs(bool)));
  connect(actionAboutRegMAS, SIGNAL(triggered()), this, SLOT(about()));
  connect(actionRegMASDocumentation, SIGNAL(triggered()), this, SLOT(showDocumentation()));
  connect(actionFitMap, SIGNAL(triggered()), mapBox, SLOT(fitInWindow()));
  connect(this, SIGNAL(resized(),mapBox, SLOT(fitInWindow()));
  connect(viewResultsButton, SIGNAL(clicked()),this, SLOT(openResults()));

  connect(&modelMainThread, SIGNAL(upgradeLogArea(const QString&)), this, SLOT(processLogArea(const QString&)));
  connect(&modelMainThread, SIGNAL(addLayerToGui(QString, QString)), this, SLOT( addLayer(QString, QString)));
  connect(layerSelector, SIGNAL(activated(int)), this, SLOT(switchToLayerFromLayerSelector(int)));
  connect(&modelMainThread, SIGNAL(updatePixelToGui(QString, int, int, QColor)), this, SLOT (updatePixel(QString, int, int, QColor)));
  connect(&modelMainThread, SIGNAL(updateImageToGui(QString, QImage)), this, SLOT (updateImage(QString, QImage)));
  connect(&modelMainThread, SIGNAL(setOutputDirNameToGui(string)), this, SLOT(setOutputDirName(string)));
  connect(&modelMainThread, SIGNAL(setGUIUnsavedStatus(bool)), this, SLOT(setUnsavedStatus(bool)));
  connect(&modelMainThread, SIGNAL(sendScenarioOptionsToGUI(const QVector<QString> &)), this, SLOT(receiveScenarioOptions(const QVector<QString> &)   ));

  // Scenario selection widget...
  scenarioWidget = new ScenarioSelectionWidget(this);
  connect(scenarioWidget->scenarioSelector, SIGNAL( activated(const QString&)), scenarioWidget, SLOT( close
()));
  connect(scenarioWidget->scenarioSelector, SIGNAL( activated(const QString&)), &modelMainThread, SLOT( retrieveScenarioNameFromGUI(const QString &)));
  //connect(scenarioWidget, SIGNAL( selectedScenarioName(const QString&)), scenarioWidget, SLOT( close()));
  //connect(scenarioWidget, SIGNAL( selectedScenarioName(const QString&)), &modelMainThread, SLOT( retrieveScenarioNameFromGUI(const QString &)));

  // Model tree viewer...
  connect(&modelMainThread, SIGNAL( treeViewerItemChangeValueToGui(string, string)  ), this, SLOT( treeViewerItemChangeValue(string, string) ));
  connect(&modelMainThread, SIGNAL( treeViewerItemRemoveToGui(string)  ), this, SLOT( treeViewerItemRemove(string) ));
  connect(&modelMainThread, SIGNAL( treeViewerAddItemToGui(string, string, string)  ), this, SLOT( treeViewerAddItem(string, string, string) ));
  connect(&modelMainThread, SIGNAL( fitInWindowToGui()), mapBox, SLOT(fitInWindow()));

  connect(mapBox, SIGNAL(  queryRequestOnPx(int, int, bool) ), &modelMainThread, SLOT ( checkQuery(int, int, bool) ) );
  connect(&modelMainThread,SIGNAL(publishQueryResults(const QString&)), pxInfoArea, SLOT (setHtml(const QString&)));
  connect(&modelMainThread,SIGNAL(activateTab(int)), tabWidget, SLOT (setCurrentIndex(int)));

  connect(&modelMainThread, SIGNAL( resetGUIForNewSimulation()  ), this, SLOT( resetGUIForNewSimulation() ) );

}

void
MainWindow::createStatusBar() {
  yearSBLabel = new QLabel(" 2000 ");
  yearSBLabel->setAlignment(Qt::AlignHCenter);
  yearSBLabel->setMinimumSize(yearSBLabel->sizeHint());

  mainSBLabel = new QLabel;
  mainSBLabel->setIndent(3);

  statusBar()->addWidget(yearSBLabel);
  statusBar()->addWidget(mainSBLabel, 1);

  yearSBLabel->setText("0");
  mainSBLabel->setText("Welcome to FFSM!");

  connect(&modelMainThread, SIGNAL(upgradeYearSBLabelToGui(const QString&)), yearSBLabel, SLOT(setText(const QString&)));
  connect(&modelMainThread, SIGNAL(upgradeMainSBLabelToGui(const QString&)), mainSBLabel, SLOT(setText(const QString&)));

}

// Manage the event of closing the application
void
MainWindow::closeEvent(QCloseEvent *event) {
  if (okToContinue()) {
    writeSettings();
    modelMainThread.stop();
    modelMainThread.wait();
    event->accept();
  } else {
    event->ignore();
    }
}

void
MainWindow::resizeEvent (QResizeEvent *event) {
  emit resized();
}


// ***************  open model / log saving functions..  ************************

void
MainWindow::setCurrentLogFileName(const QString &fileName) {
  curLogFileName = fileName;
}

void
MainWindow::setCurrentModelFileName(const QString &fileName) {
  curModelFileName = fileName;
  //setWindowModified(false);
  modelMainThread.setInputFileName(curModelFileName);

  QString shownName = "Untitled";
  if (!curModelFileName.isEmpty()) {
    shownName = strippedName(curModelFileName);
    recentFiles.removeAll(curModelFileName);
    recentFiles.prepend(curModelFileName);
    updateRecentFileActions();
  }
  setWindowTitle(tr("%2 - [%1]").arg(shownName).arg(tr("FFSM - Forest Sector Simulator")));
}

QString
MainWindow::strippedName(const QString &fullFileName) {
  return QFileInfo(fullFileName).fileName();
}

void
MainWindow::updateRecentFileActions() {
  QMutableStringListIterator i(recentFiles);
  while (i.hasNext()) {
    if (!QFile::exists(i.next()))
      i.remove();
  }

  for (int j = 0; j < MaxRecentFiles; ++j) {
    if (j < recentFiles.count()) {
      QString text = tr("&%1 %2")
            .arg(j + 1)
            .arg(strippedName(recentFiles.at(j)));
      //cerr <<text.toStdString()<<endl;
      recentFileActions[j]->setText(text);
      recentFileActions[j]->setData(recentFiles.at(j));
      recentFileActions[j]->setVisible(true);
    } else {
      recentFileActions[j]->setVisible(false);
    }
  }
  separatorAction->setVisible(!recentFiles.isEmpty());
}

bool
MainWindow::okToContinue() {
  if (modelMainThread.isRunning()) {
    int t = QMessageBox::warning(
      this,                                       // parent
      tr("FFSM"),                               // title
      tr("The model is still running.\n"   // message
        "Do you want to stop it?"),
      QMessageBox::Yes | QMessageBox::Default,    // 1st button
      QMessageBox::Cancel | QMessageBox::Escape   // 3rd button
    );
    if (t == QMessageBox::Yes) {
      modelMainThread.stop();
      modelMainThread.wait();
    } else if (t == QMessageBox::Cancel) {
      return false;
    }
  }

  if (unsavedStatus) {
    int r = QMessageBox::warning(
      this,                                       // parent
      tr("FFSM"),                               // title
      tr("The model log has not been saved.\n"   // message
        "Do you want to save it?"),
      QMessageBox::Yes ,                          // 1st button
      QMessageBox::No | QMessageBox::Default,     // 2nd button
      QMessageBox::Cancel | QMessageBox::Escape   // 3rd button
    );
    if (r == QMessageBox::Yes) {
      return save();
    } else if (r == QMessageBox::Cancel) {
      return false;
    }
  }
  return true;
}

void
MainWindow::open() {
  if (okToContinue()) {
    QString fileName = QFileDialog::getOpenFileName(
      this,
      tr("Load model file.."),
      "data/",
      tr("OpenDocument Spreadsheet (*.ods)\n" "All files (*.*)")
    );
    if (!fileName.isEmpty()){
      statusBar()->showMessage(tr("Loaded new FFSM model file"), 2000);
      setCurrentModelFileName(fileName);
      // getting the baseData path information...
      QFileInfo info(fileName);
      QString path;
      path = info.absolutePath();
      path = path+"/";
      curBaseDirectory = path;
      //modelMainThread.setBaseDirectory(curBaseDirectory);
    }
    }
}

void
MainWindow::readSettings() {
  QSettings settings("LEF", "FFSM");
  recentFiles = settings.value("recentFiles").toStringList();
  updateRecentFileActions();
}

void
MainWindow::openRecentFile() {
  if (okToContinue()) {
    QAction *action = qobject_cast<QAction *>(sender());
    if (action){
      curModelFileName=action->data().toString();
      setCurrentModelFileName(curModelFileName);
      // getting the baseData path information...
      QFileInfo info(curModelFileName);
      QString path;
      path = info.absolutePath();
      path = path+"/";
      curBaseDirectory = path;
      //modelMainThread.setBaseDirectory(curBaseDirectory);
    }
  }
}

bool
MainWindow::save() {
  if (curLogFileName.isEmpty()) {
    return saveAs();
  } else {
    cerr <<(curLogFileName.toStdString())<<endl;
    cerr <<(outputDirName.toStdString())<<endl;
    return saveLogFile(curLogFileName);
  }
  unsavedStatus = false;
  return true;
}

bool
MainWindow::saveAs() {
  QString logFileName = QFileDialog::getSaveFileName(
    this,
    tr("Save output log"),
    outputDirName,
    tr("Log files (*.log)\n" "All files (*.*)")
  );
  if (logFileName.isEmpty())
    return false;
  return saveLogFile(logFileName);
  unsavedStatus = false;
  return true;
}

bool
MainWindow::saveLogFile(const QString &logFileName) {
  QFile file(logFileName);
  if (!file.open(QIODevice::WriteOnly)) {
  QMessageBox::warning(this, tr("FFSM"),
     tr("Cannot write log file file %1:\n%2.")
    .arg(file.fileName())
    .arg(file.errorString()));
    return false;
  }
  //QString logAreaContent = logArea->toHtml();
  QString logAreaContent = logArea->toPlainText(); // Also available "toHtml()"
  QTextStream stream( &file );
  stream << logAreaContent;
  file.close();

  setCurrentLogFileName(logFileName);
  statusBar()->showMessage(tr("Log file saved"), 2000);
  unsavedStatus = false;
  return true;
}

void MainWindow::writeSettings() {
  QSettings settings("LEF", "FFSM");
    settings.setValue("recentFiles", recentFiles);
}

```

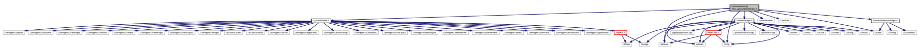
```
// ****************** Main thread controllers ***********************

void
MainWindow::startModelMainThread() {
  if (modelMainThread.isRunning()) {
    return ;
    cout <<"It seems that the model is already running..."<<endl;
    } else {
    logArea->clear();
    modelMainThread.start();
    unsavedStatus=true;
    }
}

void
MainWindow::stopModelMainThread() {
  if (! modelMainThread.isRunning()) {
    return ;
  } else {
    modelMainThread.stop();
    modelMainThread.wait();
  }
}

void
MainWindow::pauseOrResumeModelMainThread() {
  modelMainThread.pauseOrResume();
}

// ************ display px info **********************
/*void
MainWindow::sendQueryToMainThread(int px_ID){
  modelMainThread.pause();
  //modelMainThread.wait();
  modelMainThread.computeQuery(px_ID);
  modelMainThread.resume();
}*/


// ************** Map operations **********************


/**
Perform all the operation needed when adding a new layer:
 - add a layer to mapBox;
 - add the layer to layerSelector;
 - (NOTNEEDED: add the layer to layerLegend); Not needed any longer, as legend was dropped in name of the Model Status Viewer
*/
void
MainWindow::addLayer(QString layerName_h, QString layerLabel_h) {
  static int counter =0;
  mapBox->addLayer(layerName_h);
  layerSelector->addItem(layerLabel_h,layerName_h);
  // first layer added only. it is not needed as MapBox::addLayer() and QComboBox automatically switch to the new value if it is the first one :-))
  //if (counter == 0) switchToLayer(layerName_h);
  update();
  counter ++;
}

/**
Perform all the operation needed when switching layer:
 - call mapBox to switch its current layer;
 - call layerLegend to switch its layer);
I don't think it is used anywhere, but any how.. it is here...
*/
void
MainWindow::switchToLayer(QString layerName_h) {
  mapBox->switchToLayer(layerName_h);
  int index = mapBox->getLayerIndex(layerName_h);
  layerSelector->setCurrentIndex(index);
  update();
}

void
MainWindow::switchToLayerFromLayerSelector(int layerIndex_h) {
  QString layerName= layerSelector->itemData(layerIndex_h, Qt::UserRole ).toString();
  mapBox->switchToLayer(layerName);
  update();
}

void
MainWindow::updatePixel(QString layerName_h, int x_h, int y_h, QColor color_h) {
  mapBox->updatePixel(layerName_h,x_h,y_h,color_h.rgb());
  update();
}
```

```

void
MainWindow::updateImage(QString layerName_h, const QImage &image_h) {
  mapBox->updateImage(layerName_h, image_h);
  update();
}

// ************** Status viewer operations **********************
void
MainWindow::treeViewerItemChangeValue(string itemID, string newValue) {

  map<string, QTreeWidgetItem*>::iterator p;
  p=svIndex.find(itemID);
  if(p != svIndex.end())
    p->second->setText(1,newValue.c_str());
  else {
    QString tempString;
    QString tempString2 = itemID.c_str();
    tempString = "**** ERROR, Coud not change value for item "+tempString2+" in the Model Status Viewer. Item doesn't found.";
    logArea->append(tempString);
  }
  return;

}

void
MainWindow::treeViewerItemRemove(string itemID) {
  map<string, QTreeWidgetItem*>::iterator p;
  p=svIndex.find(itemID);
  if(p != svIndex.end()){
    QTreeWidgetItem *parent = p->second->parent();
    int index;
    if (parent) {
      index = parent->indexOfChild(p->second); //DONE: check if it works !!! While it should not ???? After 15 years of simulation agents should be deleted, but htey are still here in the tree.. mayme it is true it is NOT working!!! To be checken. 20071108: It works, it works.. agents are deleted when go out of the model
      delete parent->takeChild(index);
      svIndex.erase(p);
    } else {
      QString tempString = "**** ERROR, I will not delete a top level item in the Model Satus Viewer";
      logArea->append(tempString);
    }

  }
  else {
    QString tempString;
    QString tempString2 = itemID.c_str();
    tempString = "**** ERROR, Coud not delete for item "+tempString2+" in the Model Status Viewer. Item doesn't found.";
    //logArea->append(tempString); //20080111 lots of this errors when re-starting a simulation, so hidding them
  }
  return;
}

void
MainWindow::treeViewerAddItem(string text, string itemID, string parentID) {
  // searching for the parent item...
  map<string, QTreeWidgetItem*>::iterator p;
  QTreeWidgetItem *parentItem;

  p=svIndex.find(parentID);
  if(p != svIndex.end()){
    parentItem = p->second;
    QTreeWidgetItem *node = new QTreeWidgetItem(parentItem);
    svIndex.insert(pair<string, QTreeWidgetItem*>(itemID, node));
    node->setText(0, text.c_str());
  }
  else {
    QString tempString;
    QString tempString2 = itemID.c_str();
    QString tempString3 = parentID.c_str();
    tempString = "**** ERROR, Coud not add sub item "+tempString2+" to the Model Status Viewer. Parent item ("+tempString3+") doesn't found.";
    logArea->append(tempString);
  }

}

// ************** Other **********************
void
MainWindow::processLogArea(const QString& message_h){
  if(debugMsgsEnable){
    logArea->append(message_h);
  }
  else {
    if( ! message_h.startsWith("*DEBUG")){
      logArea->append(message_h);
    }
  }
}

void
MainWindow::hideDebugMsgs(bool hide){
  if(hide) debugMsgsEnable = false;
  else debugMsgsEnable = true;
}

void
MainWindow::about(){
  QMessageBox::about(this, tr("About FFSM"),
    tr("<h2>FFSM</h2>"
       "<p>Copyright &copy; 2012 Laboratoire d'Economie Forestière - LEF"
       "<br/>"
       "<p>FFSM is a flexible, spatially explicit, coupled resource and economic simulator of the Forest Sector, "
       "designed for long-term simulations of effects of government policies "
       "over different forest systems."
       "<br>It is released under the GNU GPL licence."
       "<p>For documentation and credits please refer to the project site:"
       "<br><a href=\"http://www.ffsm-project.org\">http://www.ffsm-project.org</a>"
    ));
}

void
MainWindow::showDocumentation(){
  QMessageBox::question(this, tr("FFSM Documentation"), // QMessageBox::information or QMessageBox::question
    tr("<h2>FFSM Documentation</h2>"
       "<p align=\"justify\">FFSM documentation is organised in three main categories: "
       "<p align=\"left\">(1) <b>official documentation</b> "
       "(comprising the <i>User Manual</i> and the <i>Reference Manual</i>); <br>(2) <b>contributed "
       "documentation</b> (<i>wiki</i>);<br>(3) <b>community project</b> (<i>forum</i> and <i>mailing list</i>). "
       "<p align=\"justify\">The documentation is located at "
       "<a href=\"http://www.ffsm-project.org/doc\">http://www.ffsm-project.org/doc</a>"
       "<p align=\"justify\">If you have chosen to instal a local copy of the documentation, "
       "you can access it also from the <i>Start menu</i>-><i>Programs</i>-><i>FFSM</i> "
       "(MS Windows) or directly from the following links (Linux):"
       "<br><a href=\"doc/userManual/regmasUserManual.pdf\">User Manual</a> "
       "&nbsp;-&nbsp;<a href=\"doc/referenceManual/html/index.html\">Reference Manual</a> "
       "<p>Tips:"
       "<br> - right click on a pixel to get its value across the layers;"
       "<br> - use the mouse and its wheel over the map to zoom/scroll it;"
       "</p>"
    ));
}

void
MainWindow::resetGUIForNewSimulation(){

  static int simulationCounter = 0;
  //reset map <string, QTreeWidgetItem*>       svIndex and clean the tree widget
  statusView->clear();
  map<string, QTreeWidgetItem*>::iterator p;
  //for(p=svIndex.begin(); p= svIndex.end(); p++){
    //delete p->second; // no need because they are destroyed already from statusView->clear();
  //}
  svIndex.clear();

  QTreeWidgetItem* svGeneralNode = new QTreeWidgetItem(statusView);
  svIndex.insert(pair<string, QTreeWidgetItem*>("general", svGeneralNode));
  svGeneralNode -> setText(0, "General");
  QTreeWidgetItem* svYearItem = new QTreeWidgetItem(svGeneralNode);
  svIndex.insert(pair<string, QTreeWidgetItem*>("general_year", svYearItem));
  svYearItem->setText(0, "year");
  svYearItem->setText(1, "0");
  QTreeWidgetItem* svTotalPlotsItem = new QTreeWidgetItem(svGeneralNode);
  svIndex.insert(pair<string, QTreeWidgetItem*>("general_total plots", svTotalPlotsItem));
  svTotalPlotsItem->setText(0, "total plots");
  svTotalPlotsItem->setText(1, "0");
  QTreeWidgetItem* svTotalLandItem = new QTreeWidgetItem(svGeneralNode);
  svIndex.insert(pair<string, QTreeWidgetItem*>("general_total land", svTotalLandItem));
  svTotalLandItem->setText(0, "total land");
  QTreeWidgetItem* svTotalAgrLandItem = new QTreeWidgetItem(svGeneralNode);
  svIndex.insert(pair<string, QTreeWidgetItem*>("general_total agr land", svTotalAgrLandItem));
  svTotalAgrLandItem->setText(0, "total agr land");
  QTreeWidgetItem* svOwnedAgrLandItem = new QTreeWidgetItem(svGeneralNode);
  svIndex.insert(pair<string, QTreeWidgetItem*>("general_owned agr land", svOwnedAgrLandItem));
  svOwnedAgrLandItem->setText(0, "owned agr land");
  QTreeWidgetItem* svRentedAgrLandItem = new QTreeWidgetItem(svGeneralNode);
  svIndex.insert(pair<string, QTreeWidgetItem*>("general_rented agr land", svRentedAgrLandItem));
  svRentedAgrLandItem->setText(0, "rented agr land");

  QTreeWidgetItem* svManagersNode = new QTreeWidgetItem(statusView);
  svIndex.insert(pair<string, QTreeWidgetItem*>("managers", svManagersNode));
  svManagersNode->setText(0,"Managers");

  QTreeWidgetItem* svAgentsNode = new QTreeWidgetItem(statusView);
  svIndex.insert(pair<string, QTreeWidgetItem*>("agents", svAgentsNode));
  svAgentsNode->setText(0,"Agents");

  // reset layer selector
  layerSelector->clear();
  // reset pixel info area
  pxInfoArea->setHtml("<i>Tip: Right click over a plot to retrieve its values across layers.</i>");
  // reset log area
  logArea->clear();
  // reset map

  if (simulationCounter) logArea->append("***WARNING: You are running more simulations from the GUI without closing/reopening it. It should works, but there are no guarantees. The best way is to run only one simulation from the GUI, eventually closing and opening FFSM again for further simulations.");
  simulationCounter++;

}

void
MainWindow::receiveScenarioOptions(const QVector<QString> &scenarios_h){

  //for(uint i=0;i<scenarios_h.size();i++){
  //  cout << scenarios_h.at(i).toStdString() << endl;
  //} // stange.. it works like expected !!!!

  scenarioWidget->receiveScenarioOptions(scenarios_h);
  scenarioWidget->show();
  scenarioWidget->scenarioSelector->setFocus();
  //scenarioWidget->scenarioSelector->grabMouse();
  //scenarioWidget->scenarioSelector->grabKeyboard();


}

void
MainWindow::openResults() {
  //QLabel *label = new QLabel("Hello World!");
  //label->show();
  //string aaa = curBaseDirectory.toStdString();
  //cout << "curBaseDirectory " << aaa << endl;
  //cout << "outputDirName: " << outputDirName.toStdString() << endl;
  QUrl resultsUrl(curBaseDirectory+outputDirName+"results/results.ods", QUrl::TolerantMode);
  QDesktopServices::openUrl(resultsUrl);

}
```

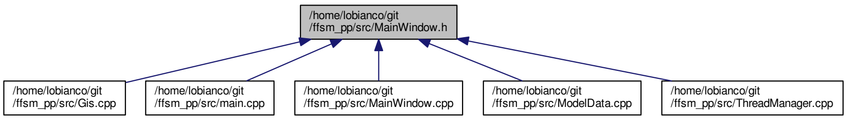
5.83 /home/lobianco/git/ffsm_pp/src/MainWindow.h File Reference

```
#include <iostream>
#include <string>
#include <sstream>
#include <QMainWindow>
#include <QTextEdit>
#include <QLabel>
#include "ui_MainWindow.h"
#include "ThreadManager.h"
#include "ScenarioSelectionWidget.h"
```

Include dependency graph for MainWindow.h:



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



Classes

- class MainWindow

  Main GUI interface.

5.84 MainWindow.h

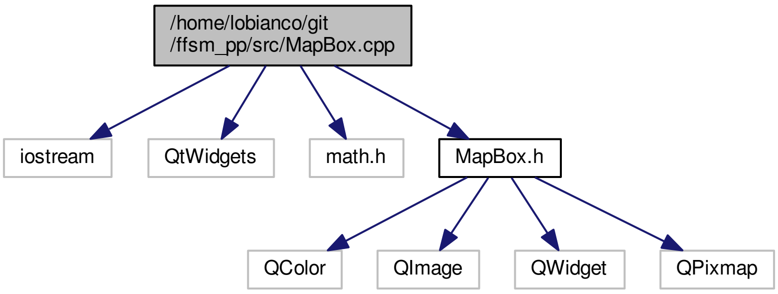
```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <iostream>
#include <string>
#include <sstream>

#include <QMainWindow>
#include <QTextEdit>
#include <QLabel>

#include "ui_MainWindow.h"

// regmas headers..
#include "ThreadManager.h"
#include "ScenarioSelectionWidget.h"

using namespace std;

//class ScenarioSelectionWidget;

/// Main GUI interface

/**
MainWindow derive from both the generic Qt QMainWindow and from Ui::MainWindow (the latter being the autmatically generated C++ code from QtDesigner).
<br>It implements code and functionality that can not be done in the QtDesigner.
*/

class MainWindow : public QMainWindow, public Ui::MainWindow {
  Q_OBJECT

public:
                      MainWindow(); ///< Constructor

  void                setCurrentLogFileName(const QString &fileName);
  void                setCurrentModelFileName(const QString &fileName);
  bool                saveLogFile(const QString &logFileName);
  QString             strippedName(const QString &fullFileName);

  QString             getModelFileName(){return curModelFileName;};
  void                setModelFileName(const QString curModelFileName_h){curModelFileName=
curModelFileName_h;};

public slots:
  void                setUnsavedStatus(bool unsavedStatus_h){unsavedStatus =
unsavedStatus_h;};
  void                setOutputDirName(string outputDirName_h){outputDirName =
outputDirName_h.c_str();};
  void                addLayer(QString layerName_h, QString layerLabel_h);
  void                switchToLayer(QString layerName_h);
  void                updatePixel(QString layerName_h, int x_h, int y_h, QColor color_h);
  void                updateImage(QString layerName_h, const QImage &image_h);
  void                switchToLayerFromLayerSelector(int layerIndex_h);
  /// Change value to an existing item in the Status Viewer
  void                treeViewerItemChangeValue(string itemID, string newValue);
  void                treeViewerItemRemove(string itemID);
  void                treeViewerAddItem(string text, string itemID, string parentID); ///< e.g.
 manager_farmer_manager agents or agent_12345_ownedHa
  void                processLogArea(const QString& message_h);
  void                resetGUIForNewSimulation(); ///< Reset the graphical elements for a new simulation
  ///// Send the request of getting the pixel info to the main thread
  //void                  sendQueryToMainThread(int px_ID);
  void                receiveScenarioOptions(const QVector<QString> &scenarios_h);


signals:
  void                currentModelFilenameChanged (QString);
  void                selectedScenarioName(const QString &scenarioName_h);
  void                resized();

protected:
  void                closeEvent(QCloseEvent *event); ///< Manage the event of closing the application
  void                resizeEvent(QResizeEvent *event); ///< Manage the event of resizing the application

private slots:
  void                open();
  bool                save();
  bool                saveAs();
  void                startModelMainThread();
  void                stopModelMainThread();
  void                pauseOrResumeModelMainThread();
  void                openRecentFile(); //already in the ui file ????
  void                hideDebugMsgs(bool hide);
  void                about();
  void                showDocumentation();
  void                openResults();

private:

  ThreadManager                modelMainThread;
  QLabel*                          yearSBLabel; ///< Status bar current year label
  QLabel*                          mainSBLabel; ///< Status bar main label
  bool                           unsavedStatus;
  QString                        outputDirName;
  QString                       curLogFileName;
  QString                     curModelFileName;
  QString                     curBaseDirectory;
  QStringList                      recentFiles;
  enum { MaxRecentFiles = 5 };
  QAction *  recentFileActions[MaxRecentFiles];
  QAction *                    separatorAction;
  bool                         debugMsgsEnable; ///< Allow debug messages to be show in the
 logArea
  ScenarioSelectionWidget      *scenarioWidget;
  /**
  Ids are based on the name of the item:
  - general
  - general_{name}
  - managers
  - manager_{managerID}
  - manager_{managerID}_{name}
  - agents
  - agent_{agentUniqueID}
  - agent_{agentUniqueID}_{name}
  */
  map <string, QTreeWidgetItem*>       svIndex; ///< Map containing the ID and the pointers to the
 status viewer

  void                         createStatusBar();
  bool                            okToContinue();
  void                            readSettings();
  void                           writeSettings();
```

```
00138   void                updateRecentFileActions();
00139 };
00140
00141 #endif
```

5.85 /home/lobianco/git/ffsm_pp/src/MapBox.cpp File Reference

```
#include <iostream>
#include <QtWidgets>
#include <math.h>
#include "MapBox.h"
```

Include dependency graph for MapBox.cpp:



Variables

- const double ZoomOutFactor = 0.8
- const double ZoomInFactor = 1 / ZoomOutFactor
- const int ScrollStep = 20

5.85.1 Variable Documentation

5.85.1.1 const int ScrollStep = 20

Definition at line 33 of file MapBox.cpp.

Referenced by MapBox::keyPressEvent().

5.85.1.2 const double ZoomInFactor = 1 / ZoomOutFactor

Definition at line 32 of file MapBox.cpp.

Referenced by MapBox::keyPressEvent(), and MapBox::wheelEvent().

5.85.1.3 const double ZoomOutFactor = 0.8

Definition at line 31 of file MapBox.cpp.

Referenced by MapBox::keyPressEvent().

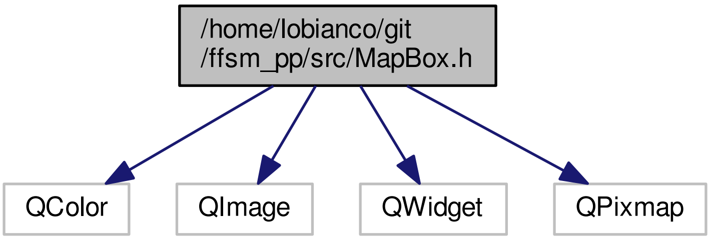
5.86 MapBox.cpp

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify *
 *   it under the terms of the GNU General Public License as published by *
 *   the Free Software Foundation; either version 3 of the License, or    *
 *   (at your option) any later version, given the compliance with the    *
 *   exceptions listed in the file COPYING that is distribued together    *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,      *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of       *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the        *
 *   GNU General Public License for more details.                         *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License    *
 *   along with this program; if not, write to the                        *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.            *
 ***************************************************************************/
#include <iostream>

//#include <QtGui>   // Qt4
#include <QtWidgets> // Qt5
#include <math.h>
#include "MapBox.h"

using namespace std;

const double ZoomOutFactor = 0.8;
const double ZoomInFactor = 1 / ZoomOutFactor;
const int ScrollStep = 20;

MapBox::MapBox(QWidget *parent):QWidget(parent) {

  currentLayerName = "";
  setCursor(Qt::CrossCursor);

  // setting source and destination init corners..
  sx1 = 0;
  sy1 = 0;
  sx2 = this->width();
  sy2 = this->height();
  dx1 = 0;
  dy1 = 0;
  dx2 = this->width();
  dy2 = this->height();
}

/**
We paint the image pixel by pixel picking up the colors from the map pointed by currentLayer.
*/
void
MapBox::paintEvent(QPaintEvent *event) {

  if (layersVector.size() < 1) return;
  QPainter painter(this);
  painter.fillRect(rect(), Qt::lightGray  );
  QPixmap pixmap = QPixmap::fromImage(currentLayer); // It doesn’t get autmoatically refreshed if I use a separate function to update the pixmap from the image
  QRectF source     (sx1, sy1, sx2-sx1, sy2-sy1); // the second point is in coordinates origin of the firt point !!!!
  QRectF destination(dx1, dy1, dx2-dx1, dy2-dy1); // the second point is in coordinates origin of the firt point !!!!
  /*
  This is the main function of the widget... the good pointa are:
  A) It takes into account the low level details of scaling, such interpolation
  B) If the destination is outside the widgets bounds, it doesn’t matter. It make its job on the widget without any error (in this sence it isnot like an array luckily...)
  */
  painter.drawPixmap(destination, pixmap, source);

}

void
MapBox::updatePixel(QString layerName_h, int x_h, int y_h, QColor color_h){
  for (uint i=0;i<layersVector.size();i++){
    if (layersNameVector.at(i) == layerName_h){
      layersVector.at(i).setPixel(x_h, y_h, color_h.rgb());
      if(layerName_h == currentLayerName){
        currentLayer = layersVector.at(i);
        update();
      }
      return;
    }
  }
}

void
MapBox::updateImage(QString layerName_h, const QImage& image_h){
  static int counter = 0;
  for (uint i=0;i<layersVector.size();i++){
    if (layersNameVector.at(i) == layerName_h){
      layersVector.at(i) = image_h;
      if(layerName_h == currentLayerName){
        currentLayer = layersVector.at(i);
        update();
      }
      if (counter == 0) { // that's the first image we got !!
        fitInWindow();
      }
      counter ++;
      return;
    }
  }
  counter ++;
  cout << "*** ERROR in MapBox::updateImage(): layerName_h "<< qPrintable(layerName_h) << " not found "<< endl;
}

void
MapBox::switchToLayer(QString layerName_h){
  if (layerName_h != currentLayerName){
    for (uint i=0;i<layersVector.size();i++){
      if (layersNameVector.at(i) == layerName_h){
        currentLayer = layersVector.at(i);
        currentLayerName = layerName_h;
        update();
        return;
      }
    }
    cout << "*** ERROR in MapBox::switchToLayer(): layerName_h "<< qPrintable(layerName_h) << " not found " << endl;
  }
}

int
MapBox::getLayerIndex(QString layerName_h){
  if( layerName_h == "") layerName_h = currentLayerName;
  for (uint i=0;i<layersVector.size();i++){
    if (layersNameVector.at(i) == layerName_h){
      return i;
    }
  }
  cout << "*** ERROR in MapBox:getLayerIndex(): layerName_h "<< qPrintable(layerName_h) << " not found "<< endl;
  return -1;
}

void
MapBox::addLayer(QString layerName_h){
  static int counter = 0;
  QImage newlayer = QImage(this->width(), this->height(), QImage::Format_RGB32);
  newlayer.fill(qRgb(255, 255, 255));
  layersVector.push_back(newlayer);
  layersNameVector.push_back(layerName_h);
  if (counter == 0) {
    currentLayerName = layerName_h;
    currentLayer = layersVector.at(0);
  }
  counter ++;
}

void
MapBox::keyPressEvent(QKeyEvent *event) {
  switch (event->key()) {
    case Qt::Key_Plus:
    zoom(ZoomInFactor);
    break;
  case Qt::Key_Minus:
    zoom(ZoomOutFactor);
    break;
  case Qt::Key_Left:
    scroll(+ScrollStep, 0);
    break;
  case Qt::Key_Right:
    scroll(-ScrollStep, 0);
    break;
  case Qt::Key_Down:
    scroll(0, -ScrollStep);
    break;
  case Qt::Key_Up:
    scroll(0, +ScrollStep);
    break;
  default:
    QWidget::keyPressEvent(event);
  }
}

void
MapBox::wheelEvent(QWheelEvent *event){
  int numDegrees = event->delta() / 8;
  double numSteps = numDegrees / 15.0f;
  zoom(pow(ZoomInFactor, numSteps));
}

void
MapBox::mousePressEvent(QMouseEvent *event){
  if (event->button() == Qt::LeftButton){
    lastDragPos = event->pos();
  }
  else if (event->button() == Qt::RightButton){
    prepareQueryEvent(event->pos());
  }
}

void
MapBox::prepareQueryEvent(QPoint click){
  double cx = ((double) click.x()); //clicked x, casted to double
  double cy = ((double) click.y()); //clicked y, casted to double
  int   mx, my = 0; // outputed x and y
  int   px_ID;  // pixel ID
  int   layerIndex = getLayerIndex();
  // checking it is not out of the destination border range..
  if (cx>dx2 || cx<dx1 || cy>dy2 || cy<dy1) return;
  mx = ( (int) (cx-dx1) * (sx2-sx1)/(dx2-dx1) + sx1); // casting to int, not round() !!
  my = ( (int) (cy-dy1) * (sy2-sy1)/(dy2-dy1) + sy1); // casting to int, not round() !!
  px_ID = mx+my*(sx2-sx1);
  emit queryRequestOnPx(px_ID, layerIndex, true);

}


void
MapBox::mouseMoveEvent(QMouseEvent *event) {
  if (event->buttons() & Qt::LeftButton) {
    scroll(event->pos().x()-lastDragPos.x(), event->pos().y()-lastDragPos.y());
    lastDragPos = event->pos();
    update();
  }
}

void MapBox::fitInWindow(){

  QPixmap pixmap = QPixmap::fromImage(currentLayer);
  double tempXScale = ( (double) this->width()) / ((double)pixmap.width());
  double tempYScale = ( (double) this->height())/ ((double)pixmap.height());

  sx1 = 0;
  sy1 = 0;
  sx2 = pixmap.width();
  sy2 = pixmap.height();
  dx1 = 0;
  dy1 = 0;

  if ( tempXScale >= tempYScale){
    dx2 = ((double)pixmap.width())*tempYScale;
    dy2 = this->height();
  } else {
    dx2 = this->width();
    dy2 = ((double)pixmap.height())*tempXScale;
  }
  update();
}

void
MapBox::zoom(double zoomFactor){
  double dx1new, dx2new, dy1new, dy2new;
  dx1new = dx2- (dx2-dx1)* ( 1+ (zoomFactor-1)/2 );
  dx2new = dx1+ (dx2-dx1)* ( 1+ (zoomFactor-1)/2 );
  dy1new = dy2- (dy2-dy1)* ( 1+ (zoomFactor-1)/2 );
  dy2new = dy1+ (dy2-dy1)* ( 1+ (zoomFactor-1)/2 );
  dx1 = dx1new;
  dy1 = dy1new;
  dx2 = dx2new;
  dy2 = dy2new;
  update();
}

void
MapBox::scroll(int deltaX, int deltaY){
  dx1 += ((double) deltaX);
  dx2 += ((double) deltaX);
  dy1 += ((double) deltaY);
  dy2 += ((double) deltaY);
  update();
}

```

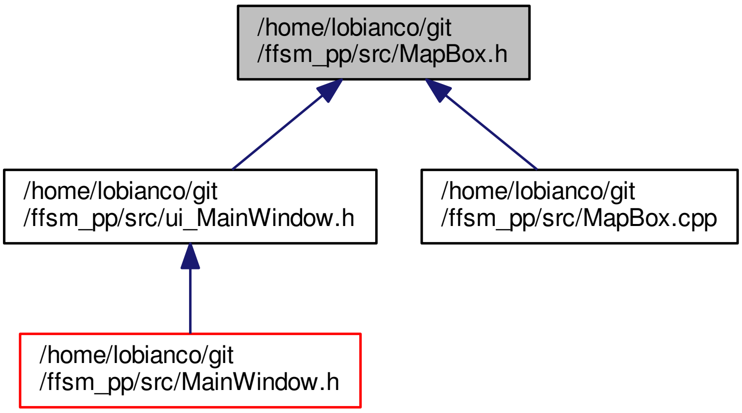
5.87 /home/lobianco/git/ffsm_pp/src/MapBox.h File Reference

```
#include <QColor>
#include <QImage>
#include <QWidget>
#include <QPixmap>
```

Include dependency graph for MapBox.h:



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



Classes

- class MapBox

  Widget to display the maps of various spacial aspects of the model.

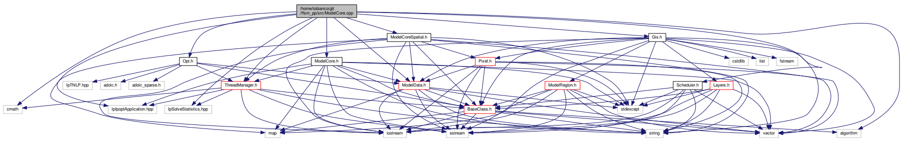
5.88 MapBox.h

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#ifndef MAPBOX_H
#define MAPBOX_H

#include <QColor> //TO.DO del
#include <QImage> //TO.DO del

#include <QWidget>
#include <QPixmap>


using namespace std;

/// Widget to display the maps of various spacial aspects of the model.

/**
This class is based on QImage. It pick-ups from layersVector the choosed layer and display it.
<br>It has methods to change the individual pixels or the whole image of a layer.
*/

class MapBox : public QWidget {
    Q_OBJECT

public:
                        MapBox(QWidget *parent = 0);
  int                   getLayerIndex(QString layerName_h=""); ///< Return the index of the specified layer (null to ask for the current one)

public slots:
  void                  updatePixel(QString layerName_h, int x_h, int y_h, QColor color_h);
  void                  updateImage(QString layerName_h, const QImage& image_h);
  void                  switchToLayer(QString layerName_h); ///< Change the layer that currentLayer and currentLayerName points
  void                  addLayer(QString layerName_h);
  void                  fitInWindow();
  void                  zoom(double zoomFactor);
  void                  scroll(int deltaX, int deltaY);

signals:
  void                  queryRequestOnPx(int px_ID, int currentLayerIndex, bool newRequest);

private:
  void                  updatePixmap(const QImage &image, bool reFit=false);
  void                  paintEvent(QPaintEvent *event); ///< Reimplementation of the standard paintEvent method.
  void                  prepareQueryEvent(QPoint click);
  void                  keyPressEvent(QKeyEvent *event);
  void                  wheelEvent(QWheelEvent *event);
  void                  mousePressEvent(QMouseEvent *event);
  void                  mouseMoveEvent(QMouseEvent *event);
  vector <QImage>                   layersVector; ///< Vector of QImages
  vector <QString>              layersNameVector; ///< Vector of layer names
  QImage                            currentLayer;
  QString                       currentLayerName;
  QPoint lastDragPos;
  double sx1, sy1, sx2, sy2; ///< coordinates of corner pixels of source - pixmap - rectangle
  double dx1, dy1, dx2, dy2; ///< coordinates of corner pixels of destination - widget - rectangle

};

#endif
```

5.89 /home/lobianco/git/ffsm_pp/src/ModelCore.cpp File Reference

```
#include <cmath>
#include <algorithm>
#include "IpIpoptApplication.hpp"
#include "IpSolveStatistics.hpp"
#include "ModelCore.h"
#include "ModelCoreSpatial.h"
#include "ModelData.h"
#include "ThreadManager.h"
#include "Opt.h"
#include "Scheduler.h"
#include "Gis.h"
```

Include dependency graph for ModelCore.cpp:



5.90 ModelCore.cpp

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d'Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#include <cmath>
#include <algorithm>

#include "IpIpoptApplication.hpp"
#include "IpSolveStatistics.hpp"

#include "ModelCore.h"
#include "ModelCoreSpatial.h"
#include "ModelData.h"
#include "ThreadManager.h"
#include "Opt.h"
#include "Scheduler.h"
#include "Gis.h"


```

```
ModelCore::ModelCore(ThreadManager* MTHREAD_h){
  MTHREAD = MTHREAD_h;
}

ModelCore::~ModelCore(){

}


/**
IMPORTANT NOTE: Volumes in Mm^3, Areas in the model in Ha (10000 m^2), in the layers in m^2
*/
void
ModelCore::runInitPeriod(){
  /**
  Some importan notes:
  V (volumes) -> at the end of the year
  In (inventary) -> at the beginning of the year
  Volumes are in Mm^3, Areas in the model in Ha (10000 m^2), in the layers in m^2
  */
  cacheSettings();              // cashe things like first year, second year, dClasses...
  initMarketModule();           // inside it uses first year, second year
  MTHREAD->DO->print();
  MTHREAD->SCD->advanceYear(); // 2005->2006
  computeInventary();           // in=f(vol_t-1)
  computeCumulativeData();      // compute cumTp_exp, vHa_exp, vHa
  runBiologicalModule();        //
  runManagementModule();
  updateMapAreas();             // update the forArea_{ft} layer on each pixel as old
 value-hArea+regArea
  MTHREAD->DO->print();
}

void
ModelCore::runSimulationYear(){
  int thisYear = MTHREAD->SCD->getYear();
  computeInventary();           // in=f(vol_t-1)
  runMarketModule();
  computeCumulativeData();      // compute cumTp_exp, vHa_exp
  runBiologicalModule();

    /*double sl = gpd("sl",11041,'softWRoundW');
    double pl = gpd("pl",11041,'softWRoundW');
    double sa = gpd("sa",11041,'softWRoundW');
    double pworld = gpd("pl", WL2,'softWRoundW');
    double st = gpd("st",11041,'softWRoundW');
    double pw = (sl*pl+sa*pworld)/st;
    cout << thisYear <<
    */

  runManagementModule();
  updateMapAreas();
  MTHREAD->DO->print();
}


void
ModelCore::initMarketModule(){
  msgOut(MSG_INFO, "Starting market module (init stage)..");
  for(uint i=0;i<regIds2.size();i++){
    int r2 = regIds2[i];

    //RPAR('pl',i,p_tr,t-1)    =  sum(p_pr, a(p_pr,p_tr)*RPAR('pl',i,p_pr,t-1))+m(i,p_tr);
    for(uint sp=0;sp<secProducts.size();sp++){
      double value = 0;
      for (uint pp=0;pp<priProducts.size();pp++){
        value += gpd("pl",r2,priProducts[pp],secondYear)*
             gpd("a",r2,priProducts[pp],secondYear,
secProducts[sp]);
      }
      value += gpd("m",r2,secProducts[sp],secondYear);
      spd(value,"pl",r2,secProducts[sp],secondYear);
    }
    // RPAR('dl',i,p_pr,t-1)    =  sum(p_tr, a(p_pr,p_tr)*RPAR('sl',i,p_tr,t-1));
    for (uint pp=0;pp<priProducts.size();pp++){
      double value=0;
      for(uint sp=0;sp<secProducts.size();sp++){
        value += gpd("sl",r2,secProducts[sp],secondYear)*
             gpd("a",r2,priProducts[pp],secondYear,
secProducts[sp]);
      }
      spd(value,"dl",r2,priProducts[pp],secondYear,true);
    }
    // RPAR('st',i,prd,t-1)     =  RPAR('sl',i,prd,t-1)+RPAR('sa',i,prd,t-1);
    // RPAR('dt',i,prd,t-1)     =  RPAR('dl',i,prd,t-1)+RPAR('da',i,prd,t-1);
    for (uint ap=0;ap<allProducts.size();ap++){
      double stvalue =   gpd("sl",r2,allProducts[ap],secondYear)
```

```
              + gpd("sa",r2,allProducts[ap],secondYear);
     double dtvalue =   gpd("dl",r2,allProducts[ap],secondYear)
              + gpd("da",r2,allProducts[ap],secondYear);
     spd(stvalue,"st",r2,allProducts[ap],secondYear,true);
     spd(dtvalue,"dt",r2,allProducts[ap],secondYear,true);
   }

   // q1(i,p_tr)  =
1/(1+((RPAR(’dl’,i,p_tr,t-1)/RPAR(’da’,i,p_tr,t-1))**(1/psi(i,p_tr)))*(RPAR(’pl’,i,p_tr,t-1)/PT(p_tr,t-1)));
   // p1(i,p_tr)              =  1-q1(i,p_tr);
   // RPAR(’dc’,i,p_tr,t-1)   =  (q1(i,p_tr)*RPAR(’da’,i,p_tr,t-1)**((psi(i,p_tr)-1)/psi(i,p_tr))+
p1(i,p_tr)*RPAR(’dl’,i,p_tr,t-1)**((psi(i,p_tr)-1)/psi(i,p_tr)))**(psi(i,p_tr)/(psi(i,p_tr)-1));
   // RPAR(’pc’,i,p_tr,t-1)   =
(RPAR(’da’,i,p_tr,t-1)/RPAR(’dc’,i,p_tr,t-1))*PT(p_tr,t-1)+(RPAR(’dl’,i,p_tr,t-1)/RPAR(’dc’,i,p_tr,t-1))*RPAR(’pl’,i,p_
   // RPAR(’pc’,i,p_pr,t-1)   =
(RPAR(’sa’,i,p_pr,t-1)/RPAR(’sc’,i,p_pr,t-1))*PT(p_pr,t-1)+(RPAR(’sl’,i,p_pr,t-1)/RPAR(’sc’,i,p_pr,t-1))*RPAR(’pl’,i,p_
   // RPAR(’pw’,i,p_tr,t-1)   =
(RPAR(’dl’,i,p_tr,t-1)*RPAR(’pl’,i,p_tr,t-1)+RPAR(’da’,i,p_tr,t-1)*PT(p_tr,t-1))/RPAR(’dt’,i,p_tr,t-1) ; //changed 201
   // K(i,p_tr,t-1)           =  k1(i,p_tr)*RPAR(’sl’,i,p_tr,t-1);
   for(uint sp=0;sp<secProducts.size();sp++){
     double psi = gpd("psi",r2,secProducts[sp],secondYear);
     double dl  = gpd("dl",r2,secProducts[sp],secondYear);
     double da  = gpd("da",r2,secProducts[sp],secondYear);
     double pl  = gpd("pl",r2,secProducts[sp],secondYear);
     double sl  = gpd("sl",r2,secProducts[sp],secondYear);
     double k1  = gpd("k1",r2,secProducts[sp],secondYear);
     double pWo = gpd("pl",WL2,secProducts[sp],secondYear); // World price
(local price for region 99999)


     double q1  = 1/ ( 1+pow(dl/da,1/psi)*(pl/pWo) );
     double p1  = 1-q1;
     double dc  = pow(
             q1*pow(da,(psi-1)/psi) + p1*pow(dl,(psi-1)/psi)
            ,
             psi/(psi-1)
            );
     double pc = (da/dc)*pWo
            +(dl/dc)*pl;
     double pw = (dl*pl+da*pWo)/(dl+da);
     double k = k1*sl;

     spd(q1,"q1",r2,secProducts[sp],firstYear,true);
     spd(p1,"p1",r2,secProducts[sp],firstYear,true);
     spd(dc,"dc",r2,secProducts[sp],secondYear,true);
     spd(pc,"pc",r2,secProducts[sp],secondYear,true);
     spd(pw,"pw",r2,secProducts[sp],secondYear,true);
     spd(k,"k",r2,secProducts[sp],secondYear,true);
   }

   // t1(i,p_pr)              =
1/(1+((RPAR(’sl’,i,p_pr,t-1)/RPAR(’sa’,i,p_pr,t-1))**(1/eta(i,p_pr)))*(RPAR(’pl’,i,p_pr,t-1)/PT(p_pr,t-1)));
   // r1(i,p_pr)              =  1-t1(i,p_pr);
   // RPAR(’sc’,i,p_pr,t-1)   =  (t1(i,p_pr)*RPAR(’sa’,i,p_pr,t-1)**((eta(i,p_pr)-1)/eta(i,p_pr))+
r1(i,p_pr)*RPAR(’sl’,i,p_pr,t-1)**((eta(i,p_pr)-1)/eta(i,p_pr)))**(eta(i,p_pr)/(eta(i,p_pr)-1))
   // RPAR(’pc’,i,p_pr,t-1)   =
(RPAR(’sa’,i,p_pr,t-1)/RPAR(’sc’,i,p_pr,t-1))*PT(p_pr,t-1)+(RPAR(’sl’,i,p_pr,t-1)/RPAR(’sc’,i,p_pr,t-1))*RPAR(’pl’,i,p_
   // RPAR(’pw’,i,p_pr,t-1)   =
(RPAR(’sl’,i,p_pr,t-1)*RPAR(’pl’,i,p_pr,t-1)+RPAR(’sa’,i,p_pr,t-1)*PT(p_pr,t-1))/RPAR(’st’,i,p_pr,t-1) ; //changed 201
   for(uint pp=0;pp<priProducts.size();pp++){

     double sl  = gpd("sl",r2,priProducts[pp],secondYear);
     double sa  = gpd("sa",r2,priProducts[pp],secondYear);
     double eta = gpd("eta",r2,priProducts[pp],secondYear);
     double pl  = gpd("pl",r2,priProducts[pp],secondYear);
     double pWo = gpd("pl",WL2,priProducts[pp],secondYear); // World price
(local price for region 99999)


     double t1 = 1/ ( 1+(pow(sl/sa,1/eta))*(pl/pWo) );
     double r1 = 1-t1;
     double sc = pow(
              t1*pow(sa,(eta-1)/eta) + r1*pow(sl,(eta-1)/eta)
           ,
              eta/(eta-1)
           );
     double pc = (sa/sc)*pWo+(sl/sc)*pl;
     double pw = (sl*pl+sa*pWo)/(sl+sa);

     spd(t1,"t1",r2,priProducts[pp],firstYear,true);
     spd(r1,"r1",r2,priProducts[pp],firstYear,true);
     spd(sc,"sc",r2,priProducts[pp],secondYear,true);
     spd(pc,"pc",r2,priProducts[pp],secondYear,true);
     spd(pw,"pw",r2,priProducts[pp],secondYear,true);
   }
 } // end of each region

```

```

  // initializing the exports to zero quantities
  for(uint r1=0;r1<l2r.size();r1++){
    for(uint r2=0;r2<l2r[r1].size();r2++){
      for(uint p=0;p<allProducts.size();p++){
       for(uint r2To=0;r2To<l2r[r1].size();r2To++){
          spd(0,"rt",l2r[r1][r2],allProducts[p],secondYear,true,
i2s(l2r[r1][r2To]));  // regional trade, it was exp in gams
        }
      }
    }
  } // end of r1 region
}

void
ModelCore::runMarketModule(){

  // *** PRE-OPTIMISATION YEARLY OPERATIONS..

  // Updating variables
  // notes:
  // - dispo_sup: not actually entering anywhere, forgiving it for now..
  // - dc0: not needed, it is just the first year demand composite
  int thisYear = MTHREAD->SCD->getYear();
  int previousYear = thisYear-1;

  for(uint i=0;i<regIds2.size();i++){
    int r2 = regIds2[i];

    // K(i,p_tr,t)  =  (1+g1(i,p_tr))*K(i,p_tr,t-1);
    // AA(i,p_tr)   =
 (sigma(p_tr)/(sigma(p_tr)+1))*RPAR(’pc’,i,p_tr,t-1)*(RPAR(’dc’,i,p_tr,t-1)**(-1/sigma(p_tr)));
    // GG(i,p_tr)   =   RPAR(’dc’,i,p_tr,t-1)*((RPAR(’pc’,i,p_tr,t-1))**(-sigma(p_tr))); //alpha
    for(uint sp=0;sp<secProducts.size();sp++){
      double g1    = gpd("g1",r2,secProducts[sp],previousYear);
      double sigma = gpd("sigma",r2,secProducts[sp]);
      double pc_1  = gpd("pc",r2,secProducts[sp],previousYear);
      double dc_1  = gpd("dc",r2,secProducts[sp],previousYear);
      double k_1   = gpd("k",r2,secProducts[sp],previousYear);

      double k  = (1+g1)*k_1;
      double aa = (sigma/(sigma+1))*pc_1*pow(dc_1,-1/sigma);
      double gg = dc_1*pow(pc_1,-sigma); //alpha

      spd(k, "k" ,r2,secProducts[sp]);
      spd(aa,"aa",r2,secProducts[sp],DATA_NOW,true);
      spd(gg,"gg",r2,secProducts[sp],DATA_NOW,true);
    }

    // BB(i,p_pr)   =
 (sigma(p_pr)/(sigma(p_pr)+1))*RPAR(’pc’,i,p_pr,t-1)*(RPAR(’sc’,i,p_pr,t-1)**(-1/sigma(p_pr)))*(In(i,p_pr,t-1)/In(i,p_p
    // FF(i,p_pr)   =
 RPAR(’sc’,i,p_pr,t-1)*((RPAR(’pc’,i,p_pr,t-1))**(-sigma(p_pr)))*(In(i,p_pr,t)/In(i,p_pr,t-1))**(gamma(p_pr)); //chi
    for(uint pp=0;pp<priProducts.size();pp++){
      double gamma = gpd("gamma",r2,priProducts[pp]);
      double sigma = gpd("sigma",r2,priProducts[pp]);
      double pc_1  = gpd("pc",r2,priProducts[pp],previousYear);
      double sc_1  = gpd("sc",r2,priProducts[pp],previousYear);
      double in    = gpd("in",r2,priProducts[pp]);
      double in_1  = gpd("in",r2,priProducts[pp],previousYear);

      double bb = (sigma/(sigma+1))*pc_1*pow(sc_1,-1/sigma)*pow(in_1/in,gamma/sigma);
      double ff = sc_1*pow(pc_1,-sigma)*pow(in/in_1,gamma); //chi

      spd(bb,"bb",r2,priProducts[pp],DATA_NOW,true);
      spd(ff,"ff",r2,priProducts[pp],DATA_NOW,true);
    }

  } // end for each region in level 2 (and updating variables)

  // *** OPTIMISATION....

  // Create an instance of the IpoptApplication
  //Opt *OPTa = new Opt(MTHREAD);
  //SmartPtr<TNLP> OPTa = new Opt(MTHREAD);
  //int initialOptYear = MTHREAD->MD->getIntSetting("initialOptYear");
  SmartPtr<IpoptApplication> application = new IpoptApplication();
  //if(thisYear == initialOptYear){
    //application = new IpoptApplication();
  //} else {
  //  application->Options()->SetStringValue("warm_start_init_point", "yes");
  //}
  string linearSolver = MTHREAD->MD->getStringSetting("linearSolver");
  application->Options()->SetStringValue("linear_solver", linearSolver); // default in ipopt is ma27
  //application->Options()->SetStringValue("hessian_approximation", "limited-memory"); // quasi-newton
 approximation of the hessian
  //application->Options()->SetIntegerValue("mumps_mem_percent", 100);
  application->Options()->SetNumericValue("obj_scaling_factor", -1); // maximisation
  application->Options()->SetNumericValue("max_cpu_time", 1800); // max 1/2 hour to find the optimus for one single year
  application->Options()->SetStringValue("check_derivatives_for_naninf", "yes"); // detect error but may crash the application.. TO.DO catch this error!
  //application->Options()->SetStringValue("nlp_scaling_method", "equilibration-based"); // much worster
  // Initialize the IpoptApplication and process the options
  ApplicationReturnStatus status;
  status = application->Initialize();
  if (status != Solve_Succeeded) {
    printf("\n\n*** Error during initialization!\n");
    msgOut(MSG_INFO,"Error during initialization! Do you have the solver compiled for the specified linear solver?");
    return;
  }


  msgOut(MSG_INFO,"Running optimisation problem for this year (it may take a few minutes for large models)..");
  status = application->OptimizeTNLP(MTHREAD->OPT);

  // *** POST OPTIMISATION....

  // post-equilibrium variables->parameters assignments..
  // RPAR(type,i,prd,t)   =  RVAR.l(type,i,prd);
  // EX(i,j,prd,t)        =  EXP.l(i,j,prd);
  // ObjT(t)              =  Obj.l ;
  // ==> in Opt::finalize_solution()

  // Retrieve some statistics about the solve
  if (status == Solve_Succeeded) {
    Index iter_count = application->Statistics()->IterationCount();
    Number final_obj = application->Statistics()->FinalObjective();
    printf("\n*** The problem solved in %d iterations!\n", iter_count);
    printf("\n*** The final value of the objective function is %e.\n", final_obj);
    msgOut(MSG_INFO, "The problem solved successfully in "+i2s(iter_count)+" iterations.")
;
    int icount = iter_count;
    double obj = final_obj;
    MTHREAD->DO->printOptLog(true, icount, obj);
  } else {
    //Number final_obj = application->Statistics()->FinalObjective();
    cout << "***ERROR: MODEL DIDN’T SOLVE FOR THIS YEAR" << endl;
        msgOut(MSG_CRITICAL_ERROR, "Model DIDN’T SOLVE for this year");
    // IMPORTANT! Don’t place the next two lines above the msgOut() function or it will crash in windows if the user press the stop button
        //Index iter_count = application->Statistics()->IterationCount(); // sys error if model didn’t solve
        //Number final_obj = application->Statistics()->FinalObjective();
        int icount = 0;
        double obj = 0;
    MTHREAD->DO->printOptLog(false, icount, obj);
  }

  for(uint r2= 0; r2<regIds2.size();r2++){  // you can use r2<=regIds2.size() to try an out-of range memory error that is not detected other than by valgrind (with a message "Invalid read of size 4 in ModelCore::runSimulationYear() in src/ModelCore.cpp:351")
    int regId = regIds2[r2];

    //  // total supply and total demand..
    //  RPAR(’st’,i,prd,t)   =  RPAR(’sl’,i,prd,t)+RPAR(’sa’,i,prd,t);
    //  RPAR(’dt’,i,prd,t)   =  RPAR(’dl’,i,prd,t)+RPAR(’da’,i,prd,t);
    //  // weighted prices.. //changed 20120419
    //  RPAR(’pw’,i,p_tr,t)    = (RPAR(’dl’,i,p_tr,t)*RPAR(’pl’,i,p_tr,t)+RPAR(’da’,i,p_tr,t)*PT(p_tr,t))/RPAR(’dt’,i,p_tr,t) ; //changed 20120419
    //  RPAR(’pw’,i,p_pr,t)    = (RPAR(’sl’,i,p_pr,t)*RPAR(’pl’,i,p_pr,t)+RPAR(’sa’,i,p_pr,t)*PT(p_pr,t))/RPAR(’st’,i,p_pr,t) ; //changed 20120419
    for (uint p=0;p<allProducts.size();p++){
      double st = gpd("sl",regId,allProducts[p])+gpd("sa",regId,allProducts[p]);
      double dt = gpd("dl",regId,allProducts[p])+gpd("da",regId,allProducts[p]);
      spd(st,"st",regId,allProducts[p]);
      spd(dt,"dt",regId,allProducts[p]);
    }
    for (uint p=0;p<secProducts.size();p++){
      double dl = gpd("dl",regId,secProducts[p]);
      double pl = gpd("pl",regId,secProducts[p]);
      double da = gpd("da",regId,secProducts[p]); // bug corrected 20120913
      double pworld = gpd("pl", WL2,secProducts[p]);
      double dt = gpd("dt",regId,secProducts[p]);
      double pw = (dl*pl+da*pworld)/dt;
      spd(pw,"pw",regId,secProducts[p]);
    }
    for (uint p=0;p<priProducts.size();p++){
      double sl = gpd("sl",regId,priProducts[p]);
      double pl = gpd("pl",regId,priProducts[p]);
```

```
      double sa = gpd("sa",regId,priProducts[p]);  // bug corrected 20120913
      double pworld = gpd("pl", WL2,priProducts[p]);
      double st = gpd("st",regId,priProducts[p]);
      double pw = (sl*pl+sa*pworld)/st;
      spd(pw,"pw",regId,priProducts[p]);
    }
  } // end of foreach region
}

void
ModelCore::runBiologicalModule(){

  msgOut(MSG_INFO, "Starting resource module..");
  hV_byPrd.clear();
  int thisYear = MTHREAD->SCD->getYear();
  int previousYear = thisYear-1;

  for(uint i=0;i<regIds2.size();i++){

    int r2 = regIds2[i];
    int regId = r2;
    // Post optimisation biological mobule..
    vector < vector < vector <double> > > hV_byPrd_regional;
    for(uint j=0;j<fTypes.size();j++){
      string ft = fTypes[j];
      vector < vector <double> > hV_byPrd_ft;

      // calculating the regeneration..
      // if we are in a year where the time of passage has not yet been reached
      // for the specific i,e,l then we use the exogenous Vregen, otherwise we
      // calculate it
      //if ( not scen("fxVreg") ,
      //  loop( (i,essence,lambda),
      //    if( ord(t)>=(tp_u1(i,essence,lambda)+2),
      //
 Vregen(i,lambda,essence,t)=regArea(i,essence,lambda,t-tp_u1(i,essence,lambda))*volHa_u1(i,essence,lambda)/1000000   ;
      //    );
      //  );
      //);
            int tp_u0 = gfd("tp",regId,ft,dClasses[0]); // time of passage to reach the first
 diameter class // bug 20140318, added ceil
      if(regType != "fixed" && (thisYear-secondYear) >= tp_u0 ) { // T.O.D.O to be checked
 -> 20121109 OK
        double pastRegArea = gfd("regArea",regId,ft,"",thisYear-tp_u0);
        double vHa = gfd("vHa",regId,ft,dClasses[1]);
        //cout << "vHa - entryVolHa: " << vHa << " / " << entryVolHa << endl;
        double vReg = pastRegArea*vHa/1000000; // T.O.D.O: check the 1000000. -> Should be ok, as area in
 ha vol in Mm^3
        sfd(vReg,"vReg",regId,ft,"");
      }

      for (uint u=0; u<dClasses.size(); u++){
        string dc = dClasses[u];
        double hr = 0;
        double pastYearVol = u?gfd("vol",regId,ft,dc,previousYear):0.;
        double hV_mort = 0.; /// \todo Harvest volumes from death trees
        vector <double> hV_byPrd_dc;

        // harvesting rate & volumes...
        //hr(u,i,essence,lambda,t) =    sum(p_pr,
 prov(u,essence,lambda,p_pr)*RPAR(’st’,i,p_pr,t)/In(i,p_pr,t));
        //hV(u,i,essence,lambda,t) = hr(u,i,essence,lambda,t) * V(u,i,lambda,essence,t-1);
        //hV_byPrd(u,i,essence,lambda,p_pr,t) =
 prov(u,essence,lambda,p_pr)*(RPAR(’st’,i,p_pr,t)/In(i,p_pr,t))*V(u,i,lambda,essence,t-1);
                //double debug =0;
        for(uint pp=0;pp<priProducts.size();pp++){
          double st = gpd("st",regId,priProducts[pp]);
          double in = gpd("in",regId,priProducts[pp]);
          double hr_pr = u?app(priProducts[pp],ft,dc)*st/ in:0;
          double hV_byPrd_dc_prd = hr_pr*pastYearVol;
          hr += hr_pr;
          hV_byPrd_dc.push_back( hV_byPrd_dc_prd);
                    //debug += hV_byPrd_dc_prd;
        }
        double hV = hr*pastYearVol;
                //double debug2 = debug-hV;

                // test passed 20131203
                //if(debug2  < -0.000000000001 || debug2  > 0.000000000001){
                //    cout << "Problems!" << endl;
                //}

        // post harvesting remained volumes computation..
        // loop(u$(ord(u)=1),
        //  first diameter class, no harvesting and fixed regenaration..
        //  V(u,i,lambda,essence,t)=(1-1/(tp(u,i,lambda,essence))-mort(u,i,lambda,essence)
 )*V(u,i,lambda,essence,t-1)
```

```
        //                         +Vregen(i,lambda,essence,t);
        // );
        // loop(u$(ord(u)>1),
        //  generic case..
        //  V(u,i,lambda,essence,t)=((1-1/(tp(u,i,lambda,essence))
        //                         -mort(u,i,lambda,essence) - hr(u,i,essence,lambda,t))*V(u,i,lambda,essence,t-1)
        // +(1/(tp(u-1,i,lambda,essence)))*beta(u,i,lambda,essence)*V(u-1,i,lambda,essence,t-1));
        double vol;
        double newMortVol;    // new mortality volumes
        double tp          = gfd("tp",regId,ft,dc);
        double mort        = u?gfd("mortCoef",regId,ft,dc):0.;
        double vReg        = gfd("vReg",regId,ft,""); // Taking it from the memory database as we could be in a fixed vReg scenario and not having calculated it from above!
        double beta        = u?gfd("betaCoef",regId,ft,dc):0.;
        //double hv2fa       = gfd("hv2fa",regId,ft,dc);
        double vHa         = gfd("vHa",regId,ft,dc);
        double finalHarvestFlag = gfd("finalHarvestFlag",regId,ft,dc);

        if(u==0){
          vol = 0.;
        } else if(u==1){
          vol = (1-1/tp-mort)*pastYearVol+vReg;
          newMortVol = mort*pastYearVol;
          double debug = vol;
        } else {
          double inc = (u==dClasses.size()-1)?0:1./tp; // we exclude the possibility for trees in the last diameter class to move to an upper class
          double tp_1 = gfd("tp",regId,ft,dClasses[u-1]);
          double pastYearVol_1 = gfd("vol",regId,ft,dClasses[u-1],previousYear);
          vol = (1-inc-mort-hr)*pastYearVol+(1/tp_1)*beta*pastYearVol_1;
          newMortVol = mort*pastYearVol;
          double debug = vol;
        }
        double freeArea_byU = u?finalHarvestFlag*1000000*hV/vHa:0; // volumes are in Mm^3, area in ha, vHa in m^3/ha
        //double debug = hv2fa*hr*pastYearVol*100;
        //cout << "regId|ft|dc| debug | freeArea: " << r2 << "|"<<ft<<"|"<<dc<<"| "<< debug << " | " << freeArea_byU << endl;

        sfd(hr,"hr",regId,ft,dc,DATA_NOW,true);
        sfd(hV,"hV",regId,ft,dc,DATA_NOW,true);
        sfd(vol,"vol",regId,ft,dc,DATA_NOW,true); // allowCreate needed for u==0
        sfd(newMortVol,"mortV",regId,ft,dc,DATA_NOW,true);

        sfd(freeArea_byU,"harvestedArea",regId,ft,dc,DATA_NOW, true);
        hV_byPrd_ft.push_back(hV_byPrd_dc);
      } // end foreach diameter classes
      hV_byPrd_regional.push_back(hV_byPrd_ft);
    } // end of each forest type
    hV_byPrd.push_back(hV_byPrd_regional);
  } // end of for each region

}

void
ModelCore::runManagementModule(){

  msgOut(MSG_INFO, "Starting management module..");
    //int thisYear = MTHREAD->SCD->getYear();
    //int previousYear = thisYear-1;
    MTHREAD->DO->expReturnsDebug.clear();
    int outputLevel = MTHREAD->MD->getIntSetting("outputLevel");
    bool weightedAverageExpectedReturns = MTHREAD->MD->getBoolSetting("weightedAverageExpectedReturns");

    //vector <vector < vector <vector <vector <double> > > > > expReturnsDebug; ///< l2_region, for type, d.c., pr prod, variable name
    //cout << "year/dc/pp/eai/cumTp/vHa/pw" << endl;

    int thisYear = MTHREAD->SCD->getYear();

  for(uint i=0;i<regIds2.size();i++){
        vector < vector <vector <vector <double> > > >  expReturnsDebug_region;

    int r2 = regIds2[i];
    int regId = r2;
    vector <double> cachedExpectedReturnsByFt;

    // PART 1: COMPUTING THE EXPECTED RETURNS FOR EACH FOREST TYPE

    for(uint j=0;j<fTypes.size();j++){
      string ft = fTypes[j];
            vector <vector <vector <double> > >  expReturnsDebug_ft;
      // Post optimisation management mobule..

```

```
     //if(regType != "fixed" && regType != "fromHrLevel"){
     // 20120910, Antonello: changed.. calculating the expected returns also for fixed and fromHrLevel regeneration (then not used but gives indication)
     // calculating the expected returns..
     //  loop ( (u,i,essence,lambda,p_pr),
     //    if (sum(u2, hV(u2,i,essence,lambda,t))= 0,
     //      expRetPondCoef(u,i,essence,lambda,p_pr) = 0;
     //    else
     //      expRetPondCoef(u,i,essence,lambda,p_pr) = hV_byPrd(u,i,essence,lambda,p_pr,t)/ sum(u2, hV(u2,i,essence,lambda,t));
     //    );
     //  );
     //  expReturns(i,essence,lambda) = sum( (u,p_pr),
     //            RPAR("pl",i,p_pr,t)*hv2fa(i,essence,lambda,u)*(1/df_byFT(u,i,lambda,essence))*// df_byFT(u,i,lambda,essence)
     //            expRetPondCoef(u,i,essence,lambda,p_pr)
     //            );
     double hV_byFT = 0.; // gfd("hV",regId,ft,DIAM_PROD); // it must include only final harvested products in order to act as weightering agent
     double expReturns = 0;


     for (uint u=0; u<dClasses.size(); u++){
       string dc = dClasses[u];
       double finalHarvestFlag = gfd("finalHarvestFlag",regId,ft,dc);
               double hV = gfd("hV",regId,ft,dc);
               hV_byFT += finalHarvestFlag * hV;
     }

           if(hV_byFT==0. || !weightedAverageExpectedReturns){ // nothing has been harvested in this pixel for this specific forest type. Let's calculate the combination product/diameter class with the highest expected return
       for (uint u=0; u<dClasses.size(); u++){
                   vector <vector <double> >  expReturnsDebug_dc;
         string dc = dClasses[u];
         double vHa              = gfd("vHa_exp",regId,ft,dc);
         double finalHarvestFlag = gfd("finalHarvestFlag",regId,ft,dc);
         double cumTp_u          = gfd("cumTp_exp",regId,ft,dc);
         for (uint pp=0;pp<priProducts.size();pp++){
                       vector <double>  expReturnsDebug_pp;
           double pw    = gpd("pw",regId,priProducts[pp]);
           double raw_amount = finalHarvestFlag*pw*vHa*app(priProducts[pp],ft,dc); // B.U.G. 20121126, it was missing app(pp,ft,dc) !!
           double anualised_amount =  MD->calculateAnnualisedEquivalent(raw_amount,cumTp_u);
                       if (anualised_amount>expReturns) {
                           expReturns=anualised_amount;
                          // if (ft == "con_highF" && regId == 11041){
                          //     cout << thisYear << "/" << dc << "/" << priProducts[pp] << "/" << anualised_amount << "/" << cumTp_u << "/" << vHa << "/" << pw << endl;
                          // }
                       }
                       if(outputLevel >= OUTVL_ALL){
                         expReturnsDebug_pp.push_back(0.0);
                         expReturnsDebug_pp.push_back(hV_byFT);
                         expReturnsDebug_pp.push_back(finalHarvestFlag);
                         expReturnsDebug_pp.push_back(0.0);
                         expReturnsDebug_pp.push_back(pw);
                         expReturnsDebug_pp.push_back(cumTp_u);
                         expReturnsDebug_pp.push_back(vHa);
                         expReturnsDebug_pp.push_back(anualised_amount);
                         expReturnsDebug_pp.push_back(0);
                       }
                       expReturnsDebug_dc.push_back(expReturnsDebug_pp);
                   } // end each pp
                   expReturnsDebug_ft.push_back(expReturnsDebug_dc);
               } // end dc
     } else {
       for (uint u=0; u<dClasses.size(); u++){
                   vector <vector <double> >  expReturnsDebug_dc;
         string dc  = dClasses[u];
         double vHa = gfd("vHa_exp",regId,ft,dc);
         double finalHarvestFlag = gfd("finalHarvestFlag",regId,ft,dc);
         double cumTp_u = gfd("cumTp_exp",regId,ft,dc);

         for (uint pp=0;pp<priProducts.size();pp++){
                       vector <double>  expReturnsDebug_pp;
           double pw    = gpd("pw",regId,priProducts[pp]);
           double pl    = gpd("pl",regId,priProducts[pp]);
           double pwor    = gpd("pl",99999,priProducts[pp]);

           double hVol_byUPp = hV_byPrd.at(i).at(j).at(u).at(pp); // harvested volumes for this product, diameter class on this region and forest type

           //double raw_amount_old = pw*hv2fa*    hVol_byUPp/hV_byFT; // old and wrong. it was in €/m^4
                       double raw_amount = finalHarvestFlag*pw*vHa* hVol_byUPp/hV_byFT; // now in €/ha if there is also thining volumes in hV_byFT, I underestimate expected returns !! TO.DO change it !! DONE,
DONE...
           /**
           see @ModelData::calculateAnnualisedEquivalent
           */
           double anualised_amount =  MD->calculateAnnualisedEquivalent(raw_amount,cumTp_u); //comTp is on diamClasses, u here is on pDiamClasses
           //cout << "reg|ft|dc|prd|raw amount|ann.amount|tp|hV|hVTot|pw|pl|pW|vHa|fHFlag;";
           //cout << regId <<";"<< ft <<";"<< dc <<";" << priProducts[pp] <<";" << raw_amount <<";"<<anualised_amount<<";";
           //cout << cumTp_u <<";"<< hVol_byUPp << ";" << hV_byFT << ";" << pw << ";" << pl << ";" << pwor << ";" << vHa << ";" << finalHarvestFlag << endl;
           expReturns += anualised_amount;

                       if(outputLevel >= OUTVL_ALL){
                         expReturnsDebug_pp.push_back(hVol_byUPp);
                         expReturnsDebug_pp.push_back(hV_byFT);
                         expReturnsDebug_pp.push_back(finalHarvestFlag);
                         expReturnsDebug_pp.push_back(finalHarvestFlag*hVol_byUPp/hV_byFT);
                         expReturnsDebug_pp.push_back(pw);
                         expReturnsDebug_pp.push_back(cumTp_u);
                         expReturnsDebug_pp.push_back(vHa);
                         expReturnsDebug_pp.push_back(MD->calculateAnnualisedEquivalent(finalHarvestFlag*pw*vHa,cumTp_u));
                         expReturnsDebug_pp.push_back(1);
                       }
                       expReturnsDebug_dc.push_back(expReturnsDebug_pp);
                   } // end for each priProducts

                   expReturnsDebug_ft.push_back(expReturnsDebug_dc);
         //expReturnsPondCoef.push_back(expReturnsPondCoef_byPrd);
               } // end for each dc
           } // ending "it has been harvested" condition
     double debug = expReturns;
     sfd(expReturns,"expReturns",regId, ft,"",DATA_NOW,true);
     cachedExpectedReturnsByFt.push_back(expReturns);
           expReturnsDebug_region.push_back(expReturnsDebug_ft);
       } // end foreach forest
       MTHREAD->DO->expReturnsDebug.push_back(expReturnsDebug_region);


   // PART 2: ALLOCATING THE HARVESTED AREA TO REGENERATION AREA BASED ON EXPECTED RETURNS

   // calculating freeArea at the end of the year and choosing the new regeneration area..
   //freeArea(i,essence,lambda) = sum(u, hv2fa(i,essence,lambda,u)*hr(u,i,essence,lambda,t)*V(u,i,lambda,essence,t-1)*100);
   //if(scen("endVreg") ,
   //  regArea(i,essence,lambda,t) = freeArea(i,essence, lambda);   // here we could introduce in/out area from other land usages
   //else
   //  loop (i,
   //    loop( (essence,lambda),
   //      if ( expReturns(i,essence,lambda) = smax( (essence2,lambda2),expReturns(i,essence2,lambda2) ),
   //        regArea (i,essence,lambda,t) =  sum( (essence2, lambda2), freeArea(i,essence2, lambda2) ) * mr;
   //      );
   //    );
   //    regArea(i,essence,lambda,t) = freeArea(i,essence, lambda)*(1-mr);   // here we could introduce in/out area from other land usages
   //  );
   double totalHarvestedArea = gfd("harvestedArea",regId,FT_ALL,DIAM_ALL);

   double maxExpReturns = *( max_element( cachedExpectedReturnsByFt.begin(), cachedExpectedReturnsByFt.end() ) );
   bool foundMaxExpReturns = false;
   for(uint j=0;j<fTypes.size();j++){
     string ft = fTypes[j];
     double harvestedAreaForThisFT = gfd("harvestedArea",regId,ft,DIAM_ALL);
     if(regType == "fixed" || regType == "fromHrLevel"){
       // regeneration area is the harvested area..
       double harvestedArea = harvestedAreaForThisFT;
       sfd(harvestedArea,"regArea",regId,ft,"",DATA_NOW,true);
     } else {
       // regeneration area is a mix between harvested area and the harvested area of te most profitting forest type..
       double regArea = 0;
       if (!foundMaxExpReturns && cachedExpectedReturnsByFt[j] == maxExpReturns){
                   // I use the foundMaxExpReturns for the unlikely event that two forest types have the same expected return to avoid overcounting of the area
         regArea += totalHarvestedArea*mr;
         foundMaxExpReturns = true;
       }
               double freq = rescaleFrequencies ? gfd("freq_norm",regId,ft,""): gfd("freq",regId,ft,""); // "probability of presence" for unmanaged forest, added 20140318
               regArea +=  harvestedAreaForThisFT*(1-mr)*freq;
       sfd(regArea,"regArea",regId,ft,"",DATA_NOW,true);
     }
   }// end of foreach forest type
   double totalRegArea = gfd("regArea",regId,FT_ALL,DIAM_ALL);
   } // end of each r2
   //vector <vector < vector <vector <vector <double> > > > > expReturnsDebug = MTHREAD->DO->expReturnsDebug;
   //cout << "bla" << endl;

}

void
ModelCore::computeInventary(){
 msgOut(MSG_INFO, "Starting computing inventary available for this year..");
 int thisYear = MTHREAD->SCD->getYear();

 // In(i,p_pr,t)   =   sum((u,lambda,essence),prov(u,essence,lambda,p_pr)*V(u,i,lambda,essence,t-1));
 for(uint i=0;i<regIds2.size();i++){
   int r2 = regIds2[i];
   for(uint pp=0;pp<priProducts.size();pp++){
     double in = 0;
     for(uint ft=0;ft<fTypes.size();ft++){
       for(uint dc=0;dc<dClasses.size();dc++){
         double vol = dc?gfd("vol",r2,fTypes[ft],dClasses[dc],thisYear-1):0.;
         in += app(priProducts[pp],fTypes[ft],dClasses[dc])*vol;
       }
     }
     spd(in,"in",r2,priProducts[pp],thisYear,true);

   }
 } // end of for each region
}

void
ModelCore::cacheSettings(){
 msgOut(MSG_INFO, "Cashing initial model settings..");
 int currentYear = MTHREAD->SCD->getYear();

 MD = MTHREAD->MD;
 firstYear = MD->getIntSetting("initialYear");
 secondYear = firstYear+1;
 thirdYear  = firstYear+2;
 WL2 = MD->getIntSetting("worldCodeLev2");
 regIds2 = MD->getRegionIds(2);
 priProducts = MD->getStringVectorSetting("priProducts");
 secProducts = MD->getStringVectorSetting("secProducts");
 allProducts = priProducts;
 allProducts.insert( allProducts.end(), secProducts.begin(), secProducts.end() );
 dClasses = MD->getStringVectorSetting("dClasses");
 pDClasses; // production diameter classes: exclude the fist diameter class below 15 cm
 pDClasses.insert(pDClasses.end(), dClasses.begin()+1, dClasses.end() );
 fTypes= MD->getForTypeIds();
 l2r = MD->getRegionIds();
 regType = MTHREAD->MD->getStringSetting("regType"); // how the regeneration should be computed (exogenous, from hr, from allocation choises)
 expType = MD->getDoubleSetting("expType");
 rescaleFrequencies = MD->getBoolSetting("rescaleFrequencies");
 if((expType<0 || expType>1) && expType != -1){
   msgOut(MSG_CRITICAL_ERROR, "expType parameter must be between 1 (expectations) and 0 (adaptative) or -1 (fixed).");
 }
 mr = MD->getDoubleSetting("mr");
}

/**
* Computing some fully exogenous parameters that require complex operations, e.g. cumulative time of passage or volume per hectare.
* This happen at the very beginning of the init period and after each simulated year
*
* It doesn't include tp and mort multipliers, but this could be added as now there is a regional versiopn of them and not just a pixel version.
*/
void
ModelCore::computeCumulativeData(){

   msgOut(MSG_INFO, "Starting computing some cumulative values..");
   int thisYear     =  MTHREAD->SCD->getYear();

   // debug
   //cout << "cumTp and vHa by dc:" << endl;
   //cout << "regId|ft|varName|0|15|25|35|45|55|65|75|85|95|150|" << endl;

   for(uint r2= 0; r2<regIds2.size();r2++){
     int regId = regIds2[r2];
     for(uint j=0;j<fTypes.size();j++){
       string ft = fTypes[j];
       // calculating the cumulative time of passage and the (cumulativelly generated) vHa for each
```

```
diameter class (depending on forest owners diam growth expectations)
        //loop(u$(ord(u)=1),
        //   cumTp(u,i,lambda,essence) = tp_u1(i,essence,lambda);
        //);
        //loop(u$(ord(u)>1),
        //   cumTp(u,i,lambda,essence) = cumTp(u-1,i,lambda,essence)+tp(u-1,i,lambda,essence);
        //);
        ////ceil(x) DNLP returns the smallest integer number greater than or equal to x
        //loop( (u,i,lambda,essence),
        //  cumTp(u,i,lambda,essence) =   ceil(cumTp(u,i,lambda,essence));
        //);
        /**
        param expType Specify how the forest owners (those that make the investments) behave will be the time of passage in the future in order to calculate the cumulative time of passage in turn used to discount future revenues.
        Will forest owners behave adaptively believing the time of passage between diameter classes will be like the observed one at time they make decision (0) or they will have full expectations believing forecasts (1) or something in the middle ?
For compatibility with the GAMS code, a -1 value means using initial simulation tp values (fixed cumTp)."
        */
                vector <double> cumTp_temp;        // cumulative time of passage to REACH a diameter class (tp is to LEAVE to the next one)
                vector <double> vHa_temp;          // volume at hectar by each diameter class [m^3/ha]
                vector <double> cumAlive_temp;     // cumulated alive rate to reach a given diameter class
                vector <double> cumTp_exp_temp;    // "expected" version of cumTp
                vector <double> vHa_exp_temp;      // "expected" version of vHa
                vector <double> cumAlive_exp_temp; // "expected" version of cumMort

        MD->setErrorLevel(MSG_NO_MSG); // as otherwise on 2007 otherwise sfd() will complain that is filling multiple years (2006 and 2007)
        for (uint u=0; u<dClasses.size(); u++){
          string dc = dClasses[u];
          double cumTp_u, cumTp_u_exp, cumTp_u_noExp, cumTp_u_fullExp;
          double vHa_u, vHa_u_exp, vHa_u_noExp, vHa_u_fullExp, beta, beta_exp, beta_noExp, beta_fullExp, mort, mort_exp, mort_noExp, mort_fullExp;
          double tp_u, tp_exp;
                    double cumAlive_u, cumAlive_exp_u;

          if(u==0) {
            // first diameter class.. expected  and real values are the same (0)
            cumTp_u = 0.;
            vHa_u    = 0.;
                        cumAlive_u = 1.;
            cumTp_temp.push_back(cumTp_u);
            cumTp_exp_temp.push_back(cumTp_u);
            vHa_temp.push_back(vHa_u);
            vHa_exp_temp.push_back(vHa_u);
                        cumAlive_temp.push_back(cumAlive_u);
                        cumAlive_exp_temp.push_back(cumAlive_u);
            sfd(cumTp_u,"cumTp",regId,ft,dc,DATA_NOW,true);
            sfd(cumTp_u,"cumTp_exp",regId,ft,dc,DATA_NOW,true);
            sfd(vHa_u,  "vHa",regId,ft,dc,DATA_NOW,true);
            sfd(vHa_u,  "vHa_exp",regId,ft,dc,DATA_NOW,true);
                        sfd(cumAlive_u,"cumAlive",regId,ft,dc,DATA_NOW,true);
                        sfd(cumAlive_u,"cumAlive_exp",regId,ft,dc,DATA_NOW,true);
          } else {
            // other diameter classes.. first dealing with real values and then with expected ones..
            // real values..
            cumTp_u = cumTp_temp[u-1] + gfd("tp",regId,ft,dClasses[u-1],thisYear); // it adds to the time of passage to reach the previous diameter class the time of passage that there should be to reach this diameter class in the year where the previous diameter class will be reached
            if (u==1){
              vHa_u = gfd("entryVolHa",regId,ft,"",thisYear);
                            mort = 0.; // not info about mortality first diameter class ("00")
            } else {
              mort = 1-pow(1-gfd("mortCoef",regId,ft,dClasses[u-1],thisYear), gfd("tp",regId,ft,dClasses[u-1],thisYear)); // mortality of the previous diameter class
                            beta = gfd("betaCoef",regId,ft,dc, thisYear);
              vHa_u = vHa_temp[u-1]*beta*(1-mort);
            }
                        cumAlive_u = max(0.,cumAlive_temp[u-1]*(1-mort));
                        cumAlive_temp.push_back(cumAlive_u);
            cumTp_temp.push_back(cumTp_u);
            vHa_temp.push_back(vHa_u);
            sfd(cumTp_u,"cumTp",regId,ft,dc,DATA_NOW,true);
            sfd(vHa_u,"vHa",regId,ft,dc,DATA_NOW,true);
                        sfd(cumAlive_u,"cumAlive",regId,ft,dc,DATA_NOW,true);

            // expected values..
            if (expType == -1){
              cumTp_u_exp = cumTp_exp_temp[u-1]+gfd("tp",regId,ft,dClasses[u-1], firstYear); // it adds to the time of passage to reach the previous diameter class the time of passage that there should be to reach this diameter class in the year where the previous diameter class will be reached
              cumTp_exp_temp.push_back(cumTp_u_exp);
              if(u==1) {
                vHa_u_exp = gfd("entryVolHa",regId,ft,"",firstYear);
```

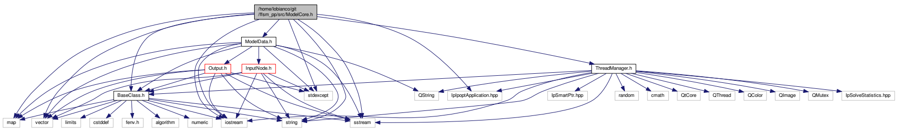
```
                                mort_exp = 0.; // not info about mortality first diameter class ("00")
              } else {
                mort_exp  = 1-pow(1-gfd("mortCoef",regId,ft,dClasses[u-1],
firstYear),gfd("tp",regId,ft,dClasses[u-1],firstYear)) ; // mortality rate of
 previous diameter class
                beta_exp  = gfd("betaCoef",regId,ft,dc, firstYear);
                vHa_u_exp = vHa_exp_temp[u-1]*beta_exp*(1-mort_exp);
              }
            } else {
              cumTp_u_noExp   = cumTp_exp_temp[u-1]+gfd("tp",regId,ft,
dClasses[u-1]);
              cumTp_u_fullExp = cumTp_exp_temp[u-1]+gfd("tp",regId,ft,
dClasses[u-1],thisYear+ceil(cumTp_exp_temp[u-1])); // it adds to the time of passage to reach the
 previous diameter class the time of passage that there should be to reach this diameter class in the year
 where the previous diameter class will be reached
              cumTp_u_exp     = cumTp_u_fullExp*expType+cumTp_u_noExp*(1-
expType);
              cumTp_exp_temp.push_back(cumTp_u_exp);
              if(u==1) {
                vHa_u_noExp   = gfd("entryVolHa",regId,ft,"",DATA_NOW);
                vHa_u_fullExp = gfd("entryVolHa",regId,ft,"",thisYear+ceil(cumTp_u));
                vHa_u_exp     = vHa_u_fullExp*expType+vHa_u_noExp*(1-
expType);
                                mort_exp      = 0. ;
              } else {
                mort_noExp   = 1-pow(1-gfd("mortCoef",regId,ft,dClasses[u-1],
DATA_NOW),cumTp_exp_temp[u]-cumTp_exp_temp[u-1]);
                mort_fullExp = 1-pow(1-gfd("mortCoef",regId,ft,dClasses[u-1],thisYear+ceil(
cumTp_temp[u-1])),cumTp_exp_temp[u]-cumTp_exp_temp[u-1]); // mortality of the previous diameter class
                beta_noExp   = gfd("betaCoef",regId,ft,dc, DATA_NOW);
                beta_fullExp = gfd("betaCoef",regId,ft,dc, thisYear+ceil(cumTp_u));
                mort_exp     = mort_fullExp*expType+mort_noExp*(1-expType);
                beta_exp     = beta_fullExp*expType+beta_noExp*(1-expType);
                vHa_u_exp    = vHa_exp_temp[u-1]*beta_exp*(1-mort_exp);
              }
            }
            vHa_exp_temp.push_back(vHa_u_exp);
                        cumAlive_exp_u = max(0.,cumAlive_exp_temp[u-1]*(1-mort_exp));
                        cumAlive_exp_temp.push_back(cumAlive_exp_u);
            sfd(cumTp_u_exp,"cumTp_exp",regId,ft,dc,DATA_NOW,true);
            sfd(vHa_u_exp,  "vHa_exp",regId,ft,dc,DATA_NOW,true);
                        sfd(cumAlive_exp_u,"cumAlive_exp",regId,ft,dc,
DATA_NOW,true);
                        //sfd(cumMort_u_exp,  "cumMort_exp",regId,ft,dc,DATA_NOW,true);

                        //cout << "********" << endl;
                        //cout << "dc: " << dClasses[u] << endl ;
                        //cout << "mort: " << mort << endl;
                        //cout << "mort_exp: " << mort_exp << endl;
                        //cout << "cumAlive: " << cumAlive_u << endl;
                        //cout << "cumAlive_exp: " << cumAlive_exp_u << endl;


          }

        } // end of each diam class


        //            // debug stuff on vHa
        //            cout << regId << "|" << ft << "|cumTp_temp|";
        //            for (uint u=0; u<dClasses.size(); u++){
        //                cout << cumTp_temp.at(u)<<"|";
        //            }
        //            cout << endl;
        //            cout << regId << "|" << ft << "|cumTp_exp|";
        //            for (uint u=0; u<dClasses.size(); u++){
        //                cout << cumTp_exp_temp.at(u)<<"|";
        //            }
        //            cout << endl;
        //            cout << regId << "|" << ft << "|vHa_temp|";
        //            for (uint u=0; u<dClasses.size(); u++){
        //                cout << vHa_temp.at(u)<<"|";
        //            }
        //            cout << endl;
        //            cout << regId << "|" << ft << "|vHa_exp|";
        //            for (uint u=0; u<dClasses.size(); u++){
        //                cout << vHa_exp_temp.at(u)<<"|";
        //            }
        //            cout << endl;


      } // end of each ft
    } // end of each region
    MD->setErrorLevel(MSG_ERROR);
}


```

```

/**
This function take for each region the difference for each forest type between the harvested area and the new regeneration one and apply such delta to each pixel of the region proportionally to the area that it already hosts.
*/
void
ModelCore::updateMapAreas(){

  msgOut(MSG_INFO, "Updating map areas..");
  map<int,double> forestArea; // foresta area by each region
  pair<int,double > forestAreaPair;
    int thisYear = MTHREAD->SCD->getYear();
  vector<int> l2Regions =  MTHREAD->MD->getRegionIds(2, true);
  vector <string> fTypes =  MTHREAD->MD->getForTypeIds();
  int nFTypes = fTypes.size();
  int nL2Regions = l2Regions.size();
  for(uint i=0;i<nL2Regions;i++){
    int regId = l2Regions[i];
    vector<Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(regId);
    ModelRegion* reg = MTHREAD->MD->getRegion(regId);
    for(uint j=0;j<nFTypes;j++){
      string ft = fTypes[j];
            double oldRegioForArea;
            if(thisYear <= firstYear+1) {
                oldRegioForArea = reg->getValue("forArea_"+ft)/10000;
            } else {
                oldRegioForArea = gfd("forArea",regId,ft,DIAM_ALL,thisYear-1);
            }
            //oldRegioForArea = reg->getValue("forArea_"+ft)/10000;
            //double debug = gfd("forArea",regId,ft,DIAM_ALL,thisYear-1);
            //double debug_diff = oldRegioForArea - debug;
            //cout << thisYear << ";" << regId << ";" << ft << ";" << oldRegioForArea << ";" << debug << ";" << debug_diff << endl;
            double harvestedArea = gfd("harvestedArea",regId,ft,DIAM_ALL); //20140206
            double regArea = gfd("regArea",regId,ft,DIAM_ALL); //20140206
      double newRegioForArea = oldRegioForArea + regArea - harvestedArea;
      sfd(newRegioForArea,"forArea",regId,ft,"",DATA_NOW, true);
      for(uint z=0;z<rpx.size();z++){
        double oldValue = rpx[z]->getDoubleValue("forArea_"+ft,true);
                double ratio = newRegioForArea/oldRegioForArea;
                double newValue = oldValue*ratio;
        rpx[z]->changeValue("forArea_"+ft, newValue);
      }

    }
  }
}



```

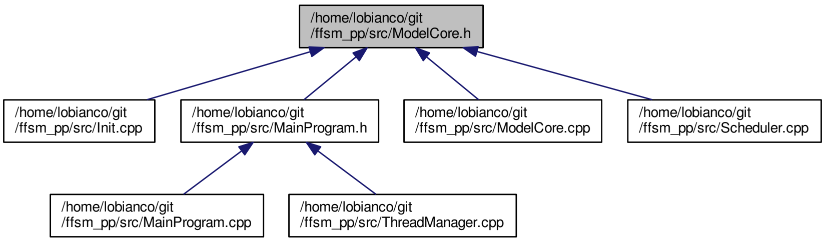
5.91 /home/lobianco/git/ffsm_pp/src/ModelCore.h File Reference

```
#include <string>
#include <vector>
#include <map>
#include <stdexcept>
#include <iostream>
#include <sstream>
#include "IpIpoptApplication.hpp"
#include "BaseClass.h"
#include "ThreadManager.h"
#include "ModelData.h"
```

Include dependency graph for ModelCore.h:



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



Classes

- class ModelCore

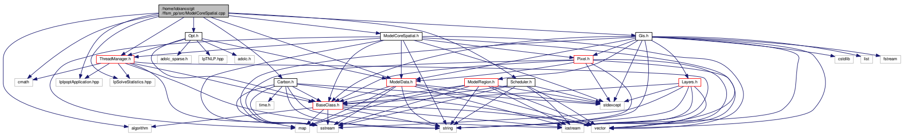
5.92 ModelCore.h

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#ifndef MODELCORE_H
#define MODELCORE_H

// Core C++ headers
#include <string>
#include <vector>
#include <map>
#include <stdexcept>
#include <iostream>
#include <sstream>

// External libraries headers
#include "IpIpoptApplication.hpp"

// Qt headers...

// RegMAS headers...
#include "BaseClass.h"
#include "ThreadManager.h"
#include "ModelData.h"

class ModelCore : public BaseClass{

public:
                        ModelCore(ThreadManager* MTHREAD_h);
                       ~ModelCore();

  void                  runInitPeriod();
  void                  runSimulationYear();

  void                  initMarketModule();       ///< computes st and pw for second year
 and several needed-only-at-t0-vars for the market module
  void                    runMarketModule();        ///< computes st (supply total) and pw (weighted price). Optimisation inside.
  void                    runBiologicalModule();    ///< computes hV, hArea and new vol at end of year
  void                    runManagementModule();    ///< computes regArea and expectedReturns

  void                    cacheSettings();          ///< just cache exogenous settings from ModelData
  void                    cachePixelExogenousData();///< computes pixel level tp, meta and mort
  void                    computeInventary();       ///< in=f(vol_t-1)
  void                    computeCumulativeData();  ///< computes cumTp, vHa, cumTp_exp, vHa_exp,
  void                    updateMapAreas();         ///< computes forArea_{ft}


private:
  // convenient handles to equivalent ModelData functions..
  double              gpd(const string &type_h, const int& regId_h, const string &prodId_h, const int& year=DATA_NOW, const string &freeDim_h="") const {return MTHREAD->MD->getProdData(type_h, regId_h, prodId_h, year, freeDim_h);};
  double              gfd(const string &type_h, const int& regId_h, const string &forType_h, const string &freeDim_h, const int& year=DATA_NOW) const {return MTHREAD->MD->getForData(type_h, regId_h, forType_h, year);};
  void                spd(const double& value_h, const string &type_h, const int& regId_h, const string &prodId_h, const int& year=DATA_NOW, const bool& allowCreate=false, const string &freeDim_h="") const {MTHREAD->MD->setProdData(value_h, type_h, regId_h, prodId_h, year, allowCreate, freeDim_h);};
  void                sfd(const double& value_h, const string &type_h, const int& regId_h, const string &forType_h, const string &freeDim_h, const int& year=DATA_NOW, const bool& allowCreate=false) const {MTHREAD->MD->setForData(value_h, type_h, regId_h, forType_h, freeDim_h, year, allowCreate);};
  bool                app(const string &prod_h, const string &forType_h, const string &dClass_h) const {return MTHREAD->MD->assessProdPossibility(prod_h, forType_h, dClass_h);};

  //vector <vector <vector <double> cumTp; /// cumulative time to reach a certain diameter class;
  //vector <vector <vector <double> vHa;   /// volumes at hectar [m^3/ha];

  ModelData* MD;
  int firstYear;
  int secondYear;
  int thirdYear;
  int WL2;
  vector <int> regIds2;
  vector <string> priProducts;
  vector <string> secProducts;
  vector <string> allProducts;
  vector <string> dClasses;
  vector <string> pDClasses;
  vector <string> fTypes;
  vector <vector <int> > l2r;
  string regType;
  double expType;
  double mr;
  vector < vector < vector < vector <double> > > > hV_byPrd; // by regId, ft, dc, pp
  //Ipopt::SmartPtr<Ipopt::IpoptApplication> application;
    bool rescaleFrequencies;


};

#endif // MODELCORE_H
```

5.93 /home/lobianco/git/ffsm_pp/src/ModelCoreSpatial.cpp File Reference

```
#include <cmath>
#include <algorithm>
#include "IpIpoptApplication.hpp"
#include "IpSolveStatistics.hpp"
#include "ModelCoreSpatial.h"
#include "ModelData.h"
#include "ThreadManager.h"
#include "Opt.h"
#include "Scheduler.h"
#include "Gis.h"
#include "Carbon.h"
```

Include dependency graph for ModelCoreSpatial.cpp:



5.94 ModelCoreSpatial.cpp

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d'Economie Forestière              *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#include <cmath>
#include <algorithm>

#include "IpIpoptApplication.hpp"
#include "IpSolveStatistics.hpp"

#include "ModelCoreSpatial.h"
#include "ModelData.h"
#include "ThreadManager.h"
#include "Opt.h"
#include "Scheduler.h"
#include "Gis.h"
#include "Carbon.h"


ModelCoreSpatial::ModelCoreSpatial(
ThreadManager *MTHREAD_h){
  MTHREAD = MTHREAD_h;
}

ModelCoreSpatial::~ModelCoreSpatial(){

}

void
ModelCoreSpatial::runInitPeriod(){
  Pixel* debug = MTHREAD->GIS->getPixel(20798);
  cacheSettings();                ///< cashe things like first year, second year, dClasses...
  initializePixelVolumes();       ///< compute px volumes vol for 2005 (including
 exogenous loaded volumes)
  assignSpMultiplierPropToVols(); // assign the spatial multiplier (used in the
 time of return) based no more on a Normal distribution but on the volumes present in the pixel: more
 volume, more the pixel is fit for the ft
  initMarketModule();             ///< inside it uses first year, second year
  initialiseDeathTimber();
  MTHREAD->DO->print();
  MTHREAD->SCD->advanceYear();    ///< 2005->2006
  int thisYear = MTHREAD->SCD->getYear(); // for debugging
  resetPixelValues();             ///< swap volumes->lagged_volumes and reset the other
 pixel vectors
  cachePixelExogenousData();      ///< compute pixel tp, meta and mort
  computeInventary();             ///< in=f(vol_t-1)
//printDebugInitRegionalValues();
  computeCumulativeData();        ///< compute cumTp_exp, vHa_exp, vHa
  initializePixelArea();          ///< compute px->area for each ft and dc (including
 exogenous loaded areas)
  runBiologicalModule();
  runManagementModule();
  MTHREAD->DO->printDebugPixelValues(); // uncomment to enable pixel-level debugging
  updateMapAreas();               ///< update the forArea_{ft} layer on each pixel as old value-hArea+regArea
  updateOtherMapData();           ///< update (if the layer exists) other gis-based data, as volumes and expected returns, taking them from the data in the px object
  sumRegionalForData();           ///< only for printing stats as forest data is never used at regional level
  initialiseCarbonModule();


  MTHREAD->DO->print();
}

void
ModelCoreSpatial::runSimulationYear(){
  int thisYear = MTHREAD->SCD->getYear(); // for debugging
  resetPixelValues();          // swap volumes->lagged_volumes and reset the other pixel vectors
  cachePixelExogenousData();   // compute pixel tp, meta and mort
  computeInventory();          // in=f(vol_t-1)
  runMarketModule();           // RUN THE MARKET OPTIMISATION HERE
  computeCumulativeData();     // compute cumTp_exp, vHa_exp
  cachePixelExogenousData();
  runBiologicalModule();
  runManagementModule();
  MTHREAD->DO->printDebugPixelValues();
  updateMapAreas();
  updateOtherMapData();        // update (if the layer exists) other gis-based data, as volumes and expected returns, taking them from the data in the px object
  sumRegionalForData();        // only for printing stats as forest data is never used at regional level
  registerCarbonEvents();
  MTHREAD->DO->print();
}

void
ModelCoreSpatial::initMarketModule(){
  msgOut(MSG_INFO, "Starting market module (init stage)..");

  for(uint i=0;i<regIds2.size();i++){
    int r2 = regIds2[i];
    //RPAR(’pl’,i,p_tr,t-1)    =  sum(p_pr, a(p_pr,p_tr)*RPAR(’pl’,i,p_pr,t-1))+m(i,p_tr);
    for(uint sp=0;sp<secProducts.size();sp++){
      double value = 0;
      for (uint pp=0;pp<priProducts.size();pp++){
        value += gpd("pl",r2,priProducts[pp],secondYear)*
        gpd("a",r2,priProducts[pp],secondYear, secProducts[sp]);
      }
      value += gpd("m",r2,secProducts[sp],secondYear);
      spd(value,"pl",r2,secProducts[sp],secondYear,true);
    }
    // RPAR(’dl’,i,p_pr,t-1)    =  sum(p_tr, a(p_pr,p_tr)*RPAR(’sl’,i,p_tr,t-1));
    for (uint pp=0;pp<priProducts.size();pp++){
      double value=0;
      for(uint sp=0;sp<secProducts.size();sp++){
        value += gpd("sl",r2,secProducts[sp],secondYear)*
        gpd("a",r2,priProducts[pp],secondYear, secProducts[sp]);
      }
      spd(value,"dl",r2,priProducts[pp],secondYear,true);
    }
    // RPAR(’st’,i,prd,t-1)     =  RPAR(’sl’,i,prd,t-1)+RPAR(’sa’,i,prd,t-1);
    // RPAR(’dt’,i,prd,t-1)     =  RPAR(’dl’,i,prd,t-1)+RPAR(’da’,i,prd,t-1);
    for (uint ap=0;ap<allProducts.size();ap++){
      //double debug = gpd("dl",r2,allProducts[ap],secondYear);
      double stvalue =   gpd("sl",r2,allProducts[ap],secondYear)
                       + gpd("sa",r2,allProducts[ap],secondYear);
      double dtvalue =   gpd("dl",r2,allProducts[ap],secondYear)
                       + gpd("da",r2,allProducts[ap],secondYear);
      spd(stvalue,"st",r2,allProducts[ap],secondYear,true);
      spd(stvalue,"stFromHarvesting",r2,allProducts[ap],secondYear,true);
      spd(dtvalue,"dt",r2,allProducts[ap],secondYear,true);
    }

    // q1(i,p_tr)   = 1/(1+((RPAR(’dl’,i,p_tr,t-1)/RPAR(’da’,i,p_tr,t-1))**(1/psi(i,p_tr)))*(RPAR(’pl’,i,p_tr,t-1)/PT(p_tr,t-1)));
    // p1(i,p_tr)                =  1-q1(i,p_tr);
    // RPAR(’dc’,i,p_tr,t-1)    =  (q1(i,p_tr)*RPAR(’da’,i,p_tr,t-1)**((psi(i,p_tr)-1)/psi(i,p_tr))+ p1(i,p_tr)*RPAR(’dl’,i,p_tr,t-1)**((psi(i,p_tr)-1)/psi(i,p_tr)))**(psi(i,p_tr)/(psi(i,p_tr)-1));
    // RPAR(’pc’,i,p_tr,t-1)    =  (RPAR(’da’,i,p_tr,t-1)/RPAR(’dc’,i,p_tr,t-1))*PT(p_tr,t-1)+(RPAR(’dl’,i,p_tr,t-1)/RPAR(’dc’,i,p_tr,t-1))*RPAR(’pl’,i,p_
    // RPAR(’pc’,i,p_pr,t-1)    =  (RPAR(’sa’,i,p_pr,t-1)/RPAR(’sc’,i,p_pr,t-1))*PT(p_pr,t-1)+(RPAR(’sl’,i,p_pr,t-1)/RPAR(’sc’,i,p_pr,t-1))*RPAR(’pl’,i,p_
    // RPAR(’pw’,i,p_tr,t-1)    =
 (RPAR('dl',i,p_tr,t-1)*RPAR('pl',i,p_tr,t-1)+RPAR('da',i,p_tr,t-1)*PT(p_tr,t-1))/RPAR('dt',i,p_tr,t-1) ; //changed 201
    // K(i,p_tr,t-1)           =  k1(i,p_tr)*RPAR('sl',i,p_tr,t-1);
    for(uint sp=0;sp<secProducts.size();sp++){
      double psi = gpd("psi",r2,secProducts[sp],secondYear);
      double dl  = gpd("dl",r2,secProducts[sp],secondYear);
      double da  = gpd("da",r2,secProducts[sp],secondYear);
      double pl  = gpd("pl",r2,secProducts[sp],secondYear);
      double sl  = gpd("sl",r2,secProducts[sp],secondYear);
      double k1  = gpd("k1",r2,secProducts[sp],secondYear);
      double pWo = gpd("pl",WL2,secProducts[sp],secondYear); // World price
 (local price for region 99999)


      double q1  = 1/ ( 1+pow(dl/da,1/psi)*(pl/pWo) );
      double p1  = 1-q1;
      double dc  = pow(
              q1*pow(da,(psi-1)/psi) + p1*pow(dl,(psi-1)/psi)
             ,
              psi/(psi-1)
             );
      double pc = (da/dc)*pWo
             +(dl/dc)*pl;
      double pw = (dl*pl+da*pWo)/(dl+da);
      double k = k1*sl;

      spd(q1,"q1",r2,secProducts[sp],firstYear,true);
      spd(p1,"p1",r2,secProducts[sp],firstYear,true);
      spd(dc,"dc",r2,secProducts[sp],secondYear,true);
      spd(pc,"pc",r2,secProducts[sp],secondYear,true);
      spd(pw,"pw",r2,secProducts[sp],secondYear,true);
      spd(k,"k",r2,secProducts[sp],secondYear,true);
    }

    // t1(i,p_pr)              =
1/(1+((RPAR('sl',i,p_pr,t-1)/RPAR('sa',i,p_pr,t-1))**(1/eta(i,p_pr)))*(RPAR('pl',i,p_pr,t-1)/PT(p_pr,t-1)));
    // r1(i,p_pr)              =  1-t1(i,p_pr);
    // RPAR('sc',i,p_pr,t-1)   =  (t1(i,p_pr)*RPAR('sa',i,p_pr,t-1)**((eta(i,p_pr)-1)/eta(i,p_pr))+
r1(i,p_pr)*RPAR('sl',i,p_pr,t-1)**((eta(i,p_pr)-1)/eta(i,p_pr)))**(eta(i,p_pr)/(eta(i,p_pr)-1))
    // RPAR('pc',i,p_pr,t-1)   =
 (RPAR('sa',i,p_pr,t-1)/RPAR('sc',i,p_pr,t-1))*PT(p_pr,t-1)+(RPAR('sl',i,p_pr,t-1)/RPAR('sc',i,p_pr,t-1))*RPAR('pl',i,p
    // RPAR('pw',i,p_pr,t-1)   =
 (RPAR('sl',i,p_pr,t-1)*RPAR('pl',i,p_pr,t-1)+RPAR('sa',i,p_pr,t-1)*PT(p_pr,t-1))/RPAR('st',i,p_pr,t-1) ; //changed 201
    for(uint pp=0;pp<priProducts.size();pp++){

      double sl  = gpd("sl",r2,priProducts[pp],secondYear);
      double sa  = gpd("sa",r2,priProducts[pp],secondYear);
      double eta = gpd("eta",r2,priProducts[pp],secondYear);
      double pl  = gpd("pl",r2,priProducts[pp],secondYear);
      double pWo = gpd("pl",WL2,priProducts[pp],secondYear); // World price
 (local price for region 99999)


      double t1 = 1/ ( 1+(pow(sl/sa,1/eta))*(pl/pWo) );
      double r1 = 1-t1;
      double sc = pow(
               t1*pow(sa,(eta-1)/eta) + r1*pow(sl,(eta-1)/eta)
            ,
               eta/(eta-1)
            );
      double pc = (sa/sc)*pWo+(sl/sc)*pl;
      double pw = (sl*pl+sa*pWo)/(sl+sa);

      spd(t1,"t1",r2,priProducts[pp],firstYear,true);
      spd(r1,"r1",r2,priProducts[pp],firstYear,true);
      spd(sc,"sc",r2,priProducts[pp],secondYear,true);
      spd(pc,"pc",r2,priProducts[pp],secondYear,true);
      spd(pw,"pw",r2,priProducts[pp],secondYear,true);
    }

    // up to here tested with gams output on 20120628, that's fine !!
  } // end for each region in level 2


  // initializing the exports to zero quantities
  // initializing  ofthe transport cost for the same region to one and distance to zero
  for(uint r1=0;r1<l2r.size();r1++){
    for(uint r2=0;r2<l2r[r1].size();r2++){
      for(uint p=0;p<allProducts.size();p++){
        for(uint r2To=0;r2To<l2r[r1].size();r2To++){
          spd(0,"rt",l2r[r1][r2],allProducts[p],secondYear,true,
i2s(l2r[r1][r2To]));  // regional trade, it was exp in gams
          if(l2r[r1][r2] == l2r[r1][r2To]){
            spd(1,"ct",l2r[r1][r2],allProducts[p],firstYear,true,
i2s(l2r[r1][r2To])); // as long this value is higher than zero, rt within the same region is not
 choosen by the solver, so the value doesn't really matters. If it is zero, the solver still works and results
 are the same, but reported rt within the region are crazy high (100000)
          }
        }
      } // end each product

      for(uint r2To=0;r2To<l2r[r1].size();r2To++){
        if(l2r[r1][r2] == l2r[r1][r2To]){
          spd(0,"dist",l2r[r1][r2],"",firstYear,true,i2s(l2r[r1][r2To])); // setting distance zero in code, so no need to put it in the data
        }
      }
    } // end of r2 regions
  } // end of r1 region
}

void
ModelCoreSpatial::runMarketModule(){
  msgOut(MSG_INFO, "Starting market module");
  static double cumOverHarvesting = 0.0;
  int thisYear     =  MTHREAD->SCD->getYear();
  int previousYear = MTHREAD->SCD->getYear()-1;

  // *** PRE-OPTIMISATION YEARLY OPERATIONS..
  for(uint i=0;i<regIds2.size();i++){
    int r2 = regIds2[i];
    for(uint sp=0;sp<secProducts.size();sp++){
      double g1      = gpd("g1",r2,secProducts[sp],previousYear);
      double sigma   = gpd("sigma",r2,secProducts[sp]);
      double pc_1    = gpd("pc",r2,secProducts[sp],previousYear);
      double dc_1    = gpd("dc",r2,secProducts[sp],previousYear);
      double k_1     = gpd("k",r2,secProducts[sp],previousYear);
      double sub_d_1 = gpd("sub_d",r2,secProducts[sp],previousYear);

      double k  = (1+g1)*k_1;
      double aa = (sigma/(sigma+1))*pc_1*pow(dc_1,-1/sigma);
      double gg = dc_1*pow(pc_1+sub_d_1,-sigma); //alpha

      spd(k, "k" ,r2,secProducts[sp]);
      spd(aa,"aa",r2,secProducts[sp],DATA_NOW,true);
      spd(gg,"gg",r2,secProducts[sp],DATA_NOW,true);
    }

    // BB(i,p_pr)   = (sigma(p_pr)/(sigma(p_pr)+1))*RPAR('pc',i,p_pr,t-1)*(RPAR('sc',i,p_pr,t-1)**(-1/sigma(p_pr)))*(In(i,p_pr,t-1)/In(i,p_p
    // FF(i,p_pr)   = RPAR('sc',i,p_pr,t-1)*((RPAR('pc',i,p_pr,t-1))**(-sigma(p_pr)))*(In(i,p_pr,t)/In(i,p_pr,t-1))**(gamma(p_pr)); //chi
    for(uint pp=0;pp<priProducts.size();pp++){
      double gamma = gpd("gamma",r2,priProducts[pp]); // elast supply to stock
      double sigma = gpd("sigma",r2,priProducts[pp]); // elast supply to price
      double sigmaCorr = sigma;
      double pc_1   = gpd("pc",r2,priProducts[pp],previousYear);
      double sc_1   = gpd("sc",r2,priProducts[pp],previousYear);
      double in     = gpd("in",r2,priProducts[pp])+gpd("in_deathTimber",r2,priProducts[pp]);
      double in_1   = gpd("in",r2,priProducts[pp],previousYear)+gpd("in_deathTimber",r2,priProducts[pp],previousYear);
      double supCorr = 1.0; // Coefficient to reduce supply function when inventory is small
      double sub_s_1 = gpd("sub_s",r2,priProducts[pp],previousYear);

      // //When inventary for a resource is almost null and further decreasing supply depends less from the price and more from the resource
      // No longer needed, but it could be used again if we face a problem where in go to zero due to too much harvesting/growth
      // //cout << "gamma orig: " << gamma << endl;
      // if (in<=0.1 && in <= in_1) { // 0.3
      //   gamma = gamma * 1.8; // 1.3: 0.65;
      //   sigmaCorr = sigma*0.2; // 0.4
      //   //supCorr = 0.7;
      //   //cout << "gamma mod: " << gamma << endl;
      // }  else if(in<=1.0 && in <= in_1){
      //   gamma = gamma * 1.8; // 1.24: 0.62;
      //   sigmaCorr = sigma*0.2; // 0.4
      //   //supCorr = 0.8;
      //   //cout << "gamma mod: " << gamma << endl;
      // }


       //if(in<=5.0){
       //  supCorr = 0.8;
       //}


      //double bb = (sigmaCorr/(sigmaCorr+1.0))*pc_1*pow(sc_1,-1.0/sigmaCorr)*pow(in_1/in,gamma/sigmaCorr);
      //double ff = sc_1*pow(pc_1,-sigmaCorr)*pow(in/in_1,gamma); //chi
      double bb = (sigmaCorr/(sigmaCorr+1.0))*pc_1*pow(sc_1,-1.0/sigmaCorr)*pow(in_1/in,gamma/sigmaCorr)*pow(1.0/supCorr,1.0/sigmaCorr);
      double ff = sc_1*pow(pc_1+sub_s_1,-sigmaCorr)*pow(in/in_1,gamma)*supCorr; //chi
      //double supCorr2 = pow(1.0/supCorr,1.0/sigmaCorr);

      spd(bb,"bb",r2,priProducts[pp],DATA_NOW,true);
      spd(ff,"ff",r2,priProducts[pp],DATA_NOW,true);
      spd(sigmaCorr,"sigmaCorr",r2,priProducts[pp],DATA_NOW,true);
      //spd(supCorr,"supCorr",r2,priProducts[pp],DATA_NOW,true);
      //spd(supCorr2,"supCorr2",r2,priProducts[pp],DATA_NOW,true);

    }

  } // end for each region in level 2 (and updating variables)



  // *** OPTIMISATION....

  // Create an instance of the IpoptApplication
  //Opt *OPTa = new Opt(MTHREAD);
  //SmartPtr<TNLP> OPTa = new Opt(MTHREAD);
  SmartPtr<IpoptApplication> application = new IpoptApplication();
  string linearSolver = MTHREAD->MD->getStringSetting("linearSolver");
  application->Options()->SetStringValue("linear_solver", linearSolver); // default in ipopt is ma27
  //application->Options()->SetStringValue("hessian_approximation", "limited-memory"); // quasi-newton approximation of the hessian
  //application->Options()->SetIntegerValue("mumps_mem_percent", 100);
  application->Options()->SetNumericValue("obj_scaling_factor", -1); // maximisation
  application->Options()->SetNumericValue("max_cpu_time", 1800); // max 1/2 hour to find the optimus for one single year
  application->Options()->SetStringValue("check_derivatives_for_naninf", "yes");

  // Initialize the IpoptApplication and process the options
  ApplicationReturnStatus status;
  status = application->Initialize();
  if (status != Solve_Succeeded) {
    printf("\n\n*** Error during initialization!\n");
    msgOut(MSG_INFO,"Error during initialization! Do you have the solver compiled for the specified linear solver?");
    return;
  }

  msgOut(MSG_INFO,"Running optimisation problem for this year (it may take a few minutes for large models)..");
  status = application->OptimizeTNLP(MTHREAD->OPT);


  // *** POST OPTIMISATION....

  // post-equilibrium variables->parameters assignments..
  // RPAR(type,i,prd,t)   =  RVAR.l(type,i,prd);
  // EX(i,j,prd,t)        =  EXP.l(i,j,prd);
  // ObjT(t)              =  Obj.l ;
  // ==> in Opt::finalize_solution()

  // Retrieve some statistics about the solve
  if (status == Solve_Succeeded) {
    Index iter_count = application->Statistics()->IterationCount();
    Number final_obj = application->Statistics()->FinalObjective();
    printf("\n*** The problem solved in %d iterations!\n", iter_count);
    printf("\n*** The final value of the objective function is %e.\n", final_obj);
    msgOut(MSG_INFO, "The problem solved successfully in "+i2s(iter_count)+" iterations.");
    int icount = iter_count;
    double obj = final_obj;
    MTHREAD->DO->printOptLog(true, icount, obj);
  } else {
    //Number final_obj = application->Statistics()->FinalObjective();
    cout << "***ERROR: MODEL DIDN’T SOLVE FOR THIS YEAR" << endl;
    msgOut(MSG_CRITICAL_ERROR, "Model DIDN’T SOLVE for this year");
    // IMPORTANT! Don’t place the next two lines above the msgOut() function or it will crash in windows if the user press the stop button
    //Index iter_count = application->Statistics()->IterationCount(); // syserror if model doesn’t solve
    //Number final_obj = application->Statistics()->FinalObjective();
    int icount = 0;
    double obj = 0;
    MTHREAD->DO->printOptLog(false, icount, obj);
  }

  for(uint r2= 0; r2<regIds2.size();r2++){  // you can use r2<=regIds2.size() to try an out-of range memory error that is not detected other than by valgrind (with a message "Invalid read of size 4 in ModelCore::runSimulationYear() in src/ModelCore.cpp:351")
    int regId = regIds2[r2];
    ModelRegion* REG = MTHREAD->MD->getRegion(regId);

    //  // total supply and total demand..
    //  RPAR(’st’,i,prd,t)   =  RPAR(’sl’,i,prd,t)+RPAR(’sa’,i,prd,t);
    //  RPAR(’dt’,i,prd,t)   =  RPAR(’dl’,i,prd,t)+RPAR(’da’,i,prd,t);
    //  // weighted prices.. //changed 20120419
    //  RPAR(’pw’,i,p_tr,t)   = (RPAR(’dl’,i,p_tr,t)*RPAR(’pl’,i,p_tr,t)+RPAR(’da’,i,p_tr,t)*PT(p_tr,t))/RPAR(’dt’,i,p_tr,t) ; //changed 20120419
    //  RPAR(’pw’,i,p_pr,t)   = (RPAR(’sl’,i,p_pr,t)*RPAR(’pl’,i,p_pr,t)+RPAR(’sa’,i,p_pr,t)*PT(p_pr,t))/RPAR(’st’,i,p_pr,t) ; //changed 20120419
    for (uint p=0;p<allProducts.size();p++){
      double st = gpd("sl",regId,allProducts[p])+gpd("sa",regId,allProducts[p]);
      double dt = gpd("dl",regId,allProducts[p])+gpd("da",regId,allProducts[p]);
      spd(st,"st",regId,allProducts[p]);
      spd(st,"st_or",regId,allProducts[p],DATA_NOW,true); // original total supply, not corrected by resetting it to min(st, inv).
      spd(dt,"dt",regId,allProducts[p]);
    }
    for (uint p=0;p<secProducts.size();p++){
      double dl = gpd("dl",regId,secProducts[p]);
      double pl = gpd("pl",regId,secProducts[p]);
      double da = gpd("da",regId,secProducts[p]); // bug corrected 20120913
      double pworld = gpd("pl", WL2,secProducts[p]);
      double dt = gpd("dt",regId,secProducts[p]);
      double pw = dt?(dl*pl+da*pworld)/dt:0.0;
      spd(pw,"pw",regId,secProducts[p]);
    }
    for (uint p=0;p<priProducts.size();p++){
      double sl = gpd("sl",regId,priProducts[p]);
      double pl = gpd("pl",regId,priProducts[p]);
      double sa = gpd("sa",regId,priProducts[p]);  // bug corrected 20120913
      double pworld = gpd("pl", WL2,priProducts[p]);
      double st = gpd("st",regId,priProducts[p]);
      double pw = st?(sl*pl+sa*pworld)/st:0.0;
      spd(pw,"pw",regId,priProducts[p]);
    }

    // Correcting st if this is over the in

    // Create a vector with all possible combinations of primary products
    vector<vector<int>> priPrCombs = MTHREAD->MD->createCombinationsVector(priProducts.size());
    int nPriPrCombs = priPrCombs.size();

    for (uint i=0;i<priPrCombs.size();i++){
      double stMkMod = 0.0;
      double sumIn = REG->inResByAnyCombination[i];
     // double sumIn2 = 0.0;
      for (uint p=0;p<priPrCombs[i].size();p++){
        stMkMod += gpd("st",regId,priProducts[priPrCombs[i][p]]);
        //sumIn2 += gpd("in",regId,priProducts[priPrCombs[i][p]]);
      }

      //if(sumIn<=0.00001){
     //   for (uint p=0;p<priPrCombs[i].size();p++){
     //     spd(0.0,"st",regId,priProducts[priPrCombs[i][p]]);
     //   }
     // } else {
      if(stMkMod>sumIn){ // if we harvested more than available
        string pProductsInvolved = "";
        for (uint p=0;p<priPrCombs[i].size();p++){
          pProductsInvolved += (priProducts[priPrCombs[i][p]]+"; ");
        }
        double inV_over_hV_ratio = stMkMod ? sumIn/stMkMod : 0.0;
        cumOverHarvesting += (stMkMod-sumIn);
        msgOut(MSG_DEBUG, "Overharvesting has happened. Year: "+i2s(thisYear)+ "Region: "+i2s(regId)+"Involved products:  "+pProductsInvolved+". sumIn: "+d2s(sumIn)+" stMkMod:" +d2s(stMkMod) + " cumOverHarvesting: "+d2s(cumOverHarvesting));
        for (uint p=0;p<priPrCombs[i].size();p++){
          double st_orig = gpd("st",regId,priProducts[priPrCombs[i][p]]);
          spd(st_orig*inV_over_hV_ratio,"st",regId,priProducts[priPrCombs[i][p]]);
        }
      }

      //}


    }

    // here we create stFromHarvesting as st - st_from_deathbiomass
    vector <double> total_st(priProducts.size(),0.);
    vector <double> in_deathTimber(priProducts.size(),0.);
    vector <double> in_aliveForest (priProducts.size(),0.);
    for (uint i=0;i<priProducts.size();i++){
      total_st[i] = gpd("st",regId,priProducts[i]);
      in_deathTimber[i] = gpd("in_deathTimber",regId,priProducts[i]);
      in_aliveForest[i] = gpd("in",regId,priProducts[i]);
    }

    vector <double> stFromHarvesting = allocateHarvesting(total_st, regId);

    for (uint i=0;i<priProducts.size();i++){
      spd(stFromHarvesting[i],"stFromHarvesting",regId,priProducts[i],
DATA_NOW,true);
    }

  } // end of each region
  if (cumOverHarvesting>0.0){
    msgOut(MSG_DEBUG, "Overharvesting is present. Year: "+i2s(thisYear)+"
 cumOverHarvesting: "+d2s(cumOverHarvesting));
  }
}


/**
 * @brief ModelCoreSpatial::runBiologicalModule
 *
 *   Changes in Area:
 *   dc     area_l  area    diff
 *   0      --------->      +regArea -areaFirstProdClass (areaMovingUp_00)
 *   15     --------->      +areaFirstPrClass -hArea_15 -areaMovingUp_15
 *   25     --------->      +areaMovingUp15 - hArea_25 - areaMovingUp_25
 *   35     --------->      +areaMovingUp25 - hArea_35 - areaMovingUp_35
 *   ...
 *   95     --------->      +areaMovingUp85 - hArea_95 - areaMovingUp_95
 *   105    --------->      +areaMovingUp95 - hArea_105
 *
 *   note: regArea is computed in the management module, not here. Further, regArea is already the net one
 of forest area changes
 */
void
ModelCoreSpatial::runBiologicalModule(){

  msgOut(MSG_INFO, "Starting resource module..");
  int thisYear = MTHREAD->SCD->getYear();
  bool useDeathTimber = MD->getBoolSetting("useDeathTimber");

  for(uint i=0;i<regIds2.size();i++){
    int r2 = regIds2[i];
    int regId = r2;
    ModelRegion* REG = MTHREAD->MD->getRegion(r2);
    //Gis* GIS = MTHREAD->GIS;
    regPx = REG->getMyPixels();
    double shareMortalityUsableTimber;
    if(useDeathTimber){
        shareMortalityUsableTimber = gfd("shareMortalityUsableTimber",r2,"","");
    } else {
        shareMortalityUsableTimber = 0.0;
    }

    for (uint p=0;p<regPx.size();p++){
      Pixel* px = regPx[p];

      double pxId = px->getID();
      //if (pxId == 3550.0){
      //    cout << "got the pixel" << endl;
      //}
      //px->expectedReturns.clear();
      for(uint j=0;j<fTypes.size();j++){
        string ft = fTypes[j];
        double pxArea_debug     = px->getDoubleValue("forArea_"+ft, true);
        vector <double>            hV_byDiam;
        vector < vector <double> > hV_byDiamAndPrd;
        vector <double> hArea_byDc;
        vector <double> newVol_byDiam;
        vector <double> vMort_byDc;
        vector <double> areasMovingUp(dClasses.size(), 0.0);
        double areaFirstProdClass;


        // A - COMPUTING THE REGENERATION..
        // if we are in a year where the time of passage has not yet been reached
        // for the specific i,e,l then we use the exogenous Vregen, otherwise we
        // calculate it
        //if ( not scen("fxVreg") ,
        //  loop( (i,essence,lambda),
        //    if( ord(t)>=(tp_u1(i,essence,lambda)+2),
        //
 Vregen(i,lambda,essence,t)=regArea(i,essence,lambda,t-tp_u1(i,essence,lambda))*volHa_u1(i,essence,lambda)/1000000   ;
        //    );
        //  );
        //);
        int tp_u0 = px->tp.at(j).at(0); // time of passage to reach the first production diameter class
 // bug 20140318, added ceil. 20140318 removed it.. model did go crazy with it
        if(thisYear == secondYear){
            px->initialDc0Area.push_back(px->area_l.at(j).at(0));
        }
        if(regType != "fixed" && (thisYear-secondYear) >= tp_u0 ) { // T.O.D.O to be
 checked -> 20121109 OK
          double pastRegArea = px->getPastRegArea(j,thisYear-tp_u0);
          double availableArea = px->area_l.at(j).at(0);
          //double entryVolHa = gfd("entryVolHa",regId,ft,"");
          double vHa = px->vHa.at(j).at(1);
          //attenction that at times could take the wrong pastRegArea if tp change too suddenly as in some "strange" scenarios
          if (oldVol2AreaMethod){
            areaFirstProdClass = pastRegArea;
          } else {
            areaFirstProdClass = min(availableArea, pastRegArea); // this is just a start and will need to include the last year area
          }
          px->vReg.push_back(areaFirstProdClass*vHa/1000000.0); // TO.DO: check the 1000000. Should be ok, as area in ha vol in Mm^3
          //if (pxId == 3550.0 && j==3){
          //    cout << "got the pixel" << endl;
          //}
          #ifdef QT_DEBUG
          if (areaFirstProdClass < 0.0){
             //msgOut(MSG_CRITICAL_ERROR,"Negative regeneration volumes in endogenous regeneration");
          }
          if ( (availableArea-pastRegArea) < -0.00000001     ){
             // in a very rare cases tp change first in a direction and then in the other, so that the wrong past regeneration area
             // is picken up.
             //msgOut(MSG_CRITICAL_ERROR,"Upgrading from dc0 more area than the available one in endogenous regeneration");
          }
          #endif
        } else {
          double regionArea = REG->getValue("forArea_"+ft,OP_SUM);
          double pxArea     = px->getDoubleValue("forArea_"+ft, true); // 20121109 bug solved (add get zero for not data)
          double regRegVolumes = gfd("vReg",r2,ft,"");
          double newVReg = regionArea ? regRegVolumes*pxArea/regionArea : 0.0;
          px->vReg.push_back(newVReg); // 20121108 BUG !!! solved // as now we have the area we could also use here entryVolHa
          // only a share of the exogenous area goes up, the regeneration one doesn't yet reach tp0:
          // areaFirstProdClass = (1.0 / px->tp.at(j).at(0) ) * px->area_l.at(j).at(0);
          areaFirstProdClass = (1.0 / ((double) tp_u0) ) * px->initialDc0Area.at(j);
          // in the exogenous period we are exogenously upgrading u0->u1 some areas but, as we do not have the regeneration
          // are corresponding to that we have also to manually add it to u0
          //px->area_l.at(j).at(0) += areaFirstProdClass;
          //areaFirstProdClass = entryVolHa ? newVReg*1000000 /entryVolHa:0.0;
          //if (pxId == 3550.0 && j==3){
          //    cout << "got the pixel" << endl;
          //}

          #ifdef QT_DEBUG
          if (areaFirstProdClass<0.0){
            // msgOut(MSG_CRITICAL_ERROR,"Negative regeneration volumes in exogenous regeneration");
          }
          if (areaFirstProdClass > px->area_l.at(j).at(0)){
             //msgOut(MSG_CRITICAL_ERROR,"Moving up area higher than available area in exogenous regeneration !");
          }
          #endif
          // vReg and entryVolHa are NOT the same thing. vReg is the yearly regeneration volumes
          // for the whole region. We can use them when we don't know the harvested area
          // entryVolHa can lead to vReg calculation only when we know the regeneration area. So in the
          // first years we use vReg and subsequently the endogenous one.
        }

        //double harvestedArea = 0;



        for (uint u=0; u<dClasses.size(); u++){
          string dc = dClasses[u];
          double hr =0;
          //double pastYearVol_reg = u ? gfd("vol",r2,ft,dc,thisYear-1): 0;
          double pastYearVol = px->vol_l.at(j).at(u);
          vector <double> hV_byPrd;
          vector <double> hr_byPrd;

          // harvesting rate & volumes...
          // hr is by region.. no reasons in one pixel the RATE of harvesting will be different than in an other pixel
          //hr(u,i,essence,lambda,t) =     sum(p_pr, prov(u,essence,lambda,p_pr)*RPAR('st',i,p_pr,t)/In(i,p_pr,t));
          //hV(u,i,essence,lambda,t) = hr(u,i,essence,lambda,t) * V(u,i,lambda,essence,t-1);
          //hV_byPrd(u,i,essence,lambda,p_pr,t) = prov(u,essence,lambda,p_pr)*(RPAR('st',i,p_pr,t)/In(i,p_pr,t))*V(u,i,lambda,essence,t-1);
          for(uint pp=0;pp<priProducts.size();pp++){
            double st = gpd("stFromHarvesting",r2,priProducts[pp]);
            double in = gpd("in",r2,priProducts[pp]);
            double hr_pr = in ? app(priProducts[pp],ft,dc)*st/in : 0.0;
            hr_byPrd.push_back( hr_pr);
            hr += hr_pr;
          }

          // adjusting for overharvesting..
          // 20160204: inserted to account that we let supply to be marginally higher than in in the mamarket module, to let the solver solving
          double origHr = hr;
          hr = min(1.0,hr);
          for(uint pp=0;pp<priProducts.size();pp++){
            double hr_pr = origHr ? hr_byPrd[pp] * min(1.0,1.0/origHr) : 0.0;
            hV_byPrd.push_back( hr_pr*pastYearVol*px->avalCoef);
          }

          double hV = hr*pastYearVol*px->avalCoef;


          hV_byDiam.push_back(hV);
          hV_byDiamAndPrd.push_back(hV_byPrd);

          // post harvesting remained volumes computation..
          // loop(u$(ord(u)=1),
          //  first diameter class, no harvesting and fixed regenaration..
          //  V(u,i,lambda,essence,t)=(1-1/(tp(u,i,lambda,essence))-mort(u,i,lambda,essence))*V(u,i,lambda,essence,t-1)
          //                           +Vregen(i,lambda,essence,t);
          // );
          // loop(u$(ord(u)>1),
          //  generic case..
          //  V(u,i,lambda,essence,t)=((1-1/(tp(u,i,lambda,essence))
          //                           -mort(u,i,lambda,essence) - hr(u,i,essence,lambda,t))*V(u,i,lambda,essence,t-1)
          //                           +(1/(tp(u-1,i,lambda,essence)))*beta(u,i,lambda,essence)*V(u-1,i,lambda,essence,t-1));
          double vol;
          double tp          = px->tp.at(j).at(u); //gfd("tp",regId,ft,dc);
          double mort        = px->mort.at(j).at(u); //gfd("mortCoef",regId,ft,dc);
          double vReg        = px->vReg.at(j); //gfd("vReg",regId,ft,""); // Taking it from the memory database as we could be in a fixed vReg scenario and not having calculated it from above!
          double beta        = px->beta.at(j).at(u); //gfd("betaCoef",regId,ft,dc);
          //double hv2fa       = gfd("hv2fa",regId,ft,dc);
          double vHa         = px->vHa.at(j).at(u); //gfd("vHa",regId,ft,dc);
          double finalHarvestFlag = gfd("finalHarvestFlag",regId,ft,dc);

          double vMort       = mort*pastYearVol;

          vMort_byDc.push_back(vMort);

          if(useDeathTimber){
            iisskey key(thisYear,r2,ft,dc);
            MD->deathTimberInventory_incrOrAdd(key,vMort*shareMortalityUsableTimber);
          }

          if(u==0){
            vol = 0.0;
          }else if(u==1){
            vol = max(0.0,(1-1/tp-mort))*pastYearVol+vReg; //Antonello, "bug" fixed 20160203: In case of very strong mortality this quantity (that doesn't include harvesting) could be negative!
            double debug = vol;
            #ifdef QT_DEBUG
            if ((1-1/tp-mort)<0.0){
              msgOut(MSG_DEBUG,"The sum of leaving trres and mortality would have lead to nevative volume if we didn't put a max. 1/tp: "+d2s(1/tp)+",  mort: "+d2s(mort)+", total coeff: "+d2s((1-1/tp-mort))+" ");
            }
            #endif
          } else {
            // time of passage and volume of smaller diameter class
            double inc = (u==dClasses.size()-1)?0:1./tp; // we exclude the possibility for trees in the last diameter class to move to an upper class
            double tp_1 = px->tp.at(j).at(u-1); //gfd("tp",regId,ft,dClasses[u-1]);
            double pastYearVol_1 = px->vol_l.at(j).at(u-1); //gfd("vol",regId,ft,dClasses[u-1],thisYear-1);
            //vol = max(0.0,(1-inc-mort-hr)*pastYearVol+(1/tp_1)*beta*pastYearVol_1);
            vol = max(0.0,(1-inc-mort)*pastYearVol-hV+(1/tp_1)*beta*pastYearVol_1); // I can't use any more hr as it is the harvesting rate over the available volumes, not the whole ones
            #ifdef QT_DEBUG
            if ((1-inc-mort)*pastYearVol-hV+(1/tp_1)*beta*pastYearVol_1 < 0){
              double realVolumes = (1-inc-mort)*pastYearVol-hV+(1/tp_1)*beta*pastYearVol_1;
              msgOut(MSG_DEBUG,"Negative real volumes ("+d2s(realVolumes)+"), possibly because of little bit larger bounds in the market module to avoid zeros. Volumes in the resource module set back to zero, so it should be ok.");
            }
            #endif
          }
          if(u != 0){ // this if is required to avoid a 0/0 and na error that then propage also in vSum()
```

```
      double inc = (u==dClasses.size()-1)?0:1.0/tp; // we exclude the possibility for trees in the last diameter class to move to an upper class
      double volumesMovingUp = inc*pastYearVol;
      double pastArea = px->area_l.at(j).at(u);

      areasMovingUp.at(u) = inc*pastArea;

      if(oldVol2AreaMethod) {
       hArea_byDc.push_back(finalHarvestFlag*1000000*hV/vHa); // volumes are in Mm^3, area in ha, vHa in m^3/ha
      } else {
       double finalHarvestedVolumes = finalHarvestFlag* hV;
       double finalHarvestedRate = pastYearVol?finalHarvestedVolumes/pastYearVol:0.0; // Here we want the harvested rate over the whole volumes, not just the available ones, so we don't need to multiply to px->avalCoef
       #ifdef QT_DEBUG
       if (finalHarvestedRate > 1.0){
        msgOut(MSG_CRITICAL_ERROR,"Negative final harvested rate.");
       }
       #endif
       hArea_byDc.push_back(finalHarvestedRate*pastArea); // volumes are in Mm^3, area in ha, vHa in m^3/ha
      }
      px->area.at(j).at(u) = max(0.0, px->area_l.at(j).at(u) - areasMovingUp.at(u) + areasMovingUp.at(u-1) - hArea_byDc.at(u));
      #ifdef QT_DEBUG
      if ((px->area_l.at(j).at(u) - areasMovingUp.at(u) + areasMovingUp.at(u-1) - hArea_byDc.at(u))< 0.0){
       msgOut(MSG_DEBUG,"If not for a max, we would have had a negative area ("+d2s(px->area_l.at(j).at(u) - areasMovingUp.at(u) + areasMovingUp.at(u-1) - hArea_byDc.at(u))+" ha).");
      }
      #endif
     } else {
      areasMovingUp.at(u) = areaFirstProdClass;
      hArea_byDc.push_back(0.);
      px->area.at(j).at(u) = px->area_l.at(j).at(u) - areasMovingUp.at(u) - hArea_byDc.at(u);
      //if (pxId == 3550.0 && j==3){
      //   cout << "got the pixel" << endl;
      //}
     }
     newVol_byDiam.push_back(vol);
     #ifdef QT_DEBUG
     if(px->area.at(j).at(u)< 0.0 || areasMovingUp.at(u) < 0.0 || hArea_byDc.at(u) < 0.0 ){
      msgOut(MSG_CRITICAL_ERROR, "Negative values in runBiologicalModel");
     }
     #endif

     //double debug = hv2fa*hr*pastYearVol*100;
     //cout << "regId|ft|dc| debug | freeArea: " << r2 << "|"<<ft<<"|"<<dc<<"| "<< debug << " | " << freeArea_byU << endl;

     //sfd(hr,"hr",regId,ft,dc);
     //sfd(hV,"hV",regId,ft,dc);
     //sfd(vol,"vol",regId,ft,dc);

     //sfd(freeArea_byU,"harvestedArea",regId,ft,dc,DATA_NOW,true);
    } // end foreach diameter classes
    px->hVol.push_back(hV_byDiam);
    px->hVol_byPrd.push_back(hV_byDiamAndPrd);
    px->hArea.push_back(hArea_byDc);
    px->vol.push_back(newVol_byDiam);
    px->vMort.push_back(vMort_byDc);


    #ifdef QT_DEBUG
    for (uint u=1; u<dClasses.size(); u++){
     double volMort = vMort_byDc[u];
     double harvVol = hV_byDiam[u];
     double vol_new = newVol_byDiam[u];
     double vol_lagged = px->vol_l.at(j).at(u);
     double gain = vol_new - (vol_lagged-harvVol-volMort);
     if (volMort > vol_lagged){
      msgOut(MSG_CRITICAL_ERROR,"mort vol > lagged volumes ?");
     }
    }
    #endif
   } // end of each forest type
  } // end of each pixel

  #ifdef QT_DEBUG
  // checking that in a region the total hVol is equal to the st for each products. 20150122 Test passed with the new availCoef
  double sumSt = 0.0;
  double sumHv = 0.0;
  for(uint pp=0;pp<priProducts.size();pp++){
      sumSt += gpd("stFromHarvesting",r2,priProducts[pp]);
    }
    for (uint p=0;p<regPx.size();p++){
      for(uint j=0;j<fTypes.size();j++){
        for (uint u=0; u<dClasses.size(); u++){
          for(uint pp=0;pp<priProducts.size();pp++){
            // by ft, dc, pp
            sumHv += regPx[p]->hVol_byPrd[j][u][pp];
          }
        }
      }
    }
    if(abs(sumSt-sumHv) > 0.000001){
      msgOut(MSG_DEBUG, "St and harvested volumes diverge in region "+REG->getRegSName()+". St: "+d2s(sumSt)+" hV: "+d2s(sumHv));
    }
    #endif
  } // end of each region

}



void
ModelCoreSpatial::runManagementModule(){
  msgOut(MSG_INFO, "Starting management module..");
  vector<string> allFTypes = MTHREAD->MD->getForTypeIds(true);
  map<string,double> hAreaByFTypeGroup = vectorToMap(allFTypes,0.0);
  int thisYear = MTHREAD->SCD->getYear();

  // Post optimisation management mobule..
  for(uint i=0;i<regIds2.size();i++){
    int r2 = regIds2[i];
    int regId = r2;
    ModelRegion* REG = MTHREAD->MD->getRegion(r2);
    regPx = REG->getMyPixels();

    // Dealing with area change..
    double fArea_reg     = REG->getArea();
    double fArea_diff    = 0.0;
    double fArea_reldiff = 0.0;
    if(forestAreaChangeMethod=="relative"){
      fArea_reldiff = gfd("forestChangeAreaIncrementsRel",r2,"","",DATA_NOW);
      fArea_diff    = fArea_reg * fArea_reldiff;
    } else if (forestAreaChangeMethod=="absolute"){
      fArea_diff    = gfd("forestChangeAreaIncrementsHa",r2,"","",DATA_NOW);
      //fArea_reldiff = fArea_diff / fArea_reg;
    }
    double regHArea = 0.0; // for the warning




    for (uint p=0;p<regPx.size();p++){
      Pixel* px = regPx[p];
      px->expectedReturns.clear();
      px->expectedReturnsNotCorrByRa.clear(); // BUG discovered 20160825
      resetMapValues(hAreaByFTypeGroup,0.0);
      double totalHarvestedArea = vSum(px->hArea); // still need to remove the forest decrease areas..
      vector<double> thisYearRegAreas(fTypes.size(),0.0); // initialize a vector of fTypes.size() zeros.
      vector<double> expectedReturns(fTypes.size(),0.0);  // uncorrected expected returns (without considering transaction costs). These are in form of eai

      double fArea_px = vSum(px->area);
      double fArea_diff_px = fArea_px * fArea_diff/ fArea_reg;
      double fArea_incr = max(0.0,fArea_diff_px);
      double fArea_decr = - min(0.0,fArea_diff_px);
      double fArea_decr_rel = totalHarvestedArea?min(1.0,fArea_decr/totalHarvestedArea):0.0;
      regHArea += totalHarvestedArea;
      totalHarvestedArea = totalHarvestedArea *(1-fArea_decr_rel);


      // A - Computing the harvestingArea by parent ft group (for the allocation according to the prob of presence):
      for(uint j=0;j<fTypes.size();j++){
        string ft = fTypes[j];
        string parentFt = MTHREAD->MD->getForTypeParentId(ft);
        double hAreaThisFt=vSum(px->hArea.at(j))*(1-fArea_decr_rel);
        incrMapValue(hAreaByFTypeGroup,parentFt,hAreaThisFt); // increment the parent ft of the harvested area, neeed for assigning the frequences (prob. of presence)
      }

      // B - Computing the uncorrected expected returns (without considering transaction costs)
      // 20120910, Antonello: changed.. calculating the expected returns also for fixed and fromHrLevel regeneration (then not used but gives indication)
      // calculating the expected returns..
      //  loop ( (u,i,essence,lambda,p_pr),
      //    if (sum(u2, hV(u2,i,essence,lambda,t))= 0,
      //      expRetPondCoef(u,i,essence,lambda,p_pr) = 0;
      //    else
      //      expRetPondCoef(u,i,essence,lambda,p_pr) = hV_byPrd(u,i,essence,lambda,p_pr,t)/ sum(u2, hV(u2,i,essence,lambda,t));
      //    );
      //  );
      //  expReturns(i,essence,lambda) = sum( (u,p_pr),
      //            RPAR("pl",i,p_pr,t)*hv2fa(i,essence,lambda,u)*(1/df_byFT(u,i,lambda,essence))*// df_byFT(u,i,lambda,essence)
      //            expRetPondCoef(u,i,essence,lambda,p_pr)
      //            );
      for(uint j=0;j<fTypes.size();j++){
        string ft = fTypes[j];
        double expReturns = 0.;
        int optDc = 0; // "optimal diameter class", the one on which the expected returns are computed
        for (uint u=0; u<dClasses.size(); u++){
          string dc = dClasses[u];
          double vHa         = px->vHa_exp.at(j).at(u);
          double finalHarvestFlag = gfd("finalHarvestFlag",regId,ft,dc);
          double cumTp_u = px->cumTp_exp.at(j).at(u);
          for (uint pp=0;pp<priProducts.size();pp++){
            double pl             = gpd("pl",regId,priProducts[pp]); // note that this is the OBSERVED price. If we call it at current year+cumTp_u we would have the expected price. But we would first have to compute it, as pw is weigthed price world-local and we don't have local price for the future. DONE 20141202 ;-)
            double worldCurPrice = gpd("pl",WL2,priProducts[pp]);
            double worldFutPrice = gpd("pl",WL2,priProducts[pp],thisYear+cumTp_u);
            double sl             = gpd("sl",regId,priProducts[pp]);
            double sa             = gpd("sa",regId,priProducts[pp]);
            double pw_exp         = computeExpectedPrice(pl, worldCurPrice, worldFutPrice, sl, sa, px->expTypePrices); //20141030: added the expected price!
            double raw_amount = finalHarvestFlag*pw_exp*vHa*app(priProducts[pp],ft,dc); // B.U.G. 20121126, it was missing app(pp,ft,dc) !!
            double anualised_amount =  MD->calculateAnnualisedEquivalent(raw_amount,cumTp_u);
            if (anualised_amount>expReturns) {
              expReturns=anualised_amount;
              optDc = u;
            }
          }
        }
        px->expectedReturnsNotCorrByRa.push_back(expReturns);
        if(MD->getBoolSetting("heterogeneousRiskAversion")){
          double ra = px->getDoubleValue("ra");
          double cumMort = 1-px->cumAlive_exp.at(j).at(optDc);
          //cout << px->getID() << "\t" << ft << "\t\t" << "optDc" << optDc << "\t" << cumMort << endl;
          double origExpReturns = expReturns;
          expReturns = origExpReturns * (1.0 - ra*cumMort);
        }
        px->expectedReturns.push_back(expReturns);
        expectedReturns.at(j) = expReturns;
      } // end foreach forest type

      for(uint j=0;j<fTypes.size();j++){
        string ft = fTypes[j];
        forType* thisFt = MTHREAD->MD->getForType(ft);

        double harvestedAreaForThisFT = vSum(px->hArea.at(j))*(1-fArea_decr_rel); // gfd("harvestedArea",regId,ft,DIAM_ALL);
        vector<double> corrExpectedReturns(fTypes.size(),0.0); //  corrected expected returns (considering transaction costs). These are in form of  NPV

        // C - Computing the corrected expected returns including transaction costs
        for(uint j2=0;j2<fTypes.size();j2++){
          string ft2 = fTypes[j2];
          double invTransCost = gfd("invTransCost",regId,ft,ft2,DATA_NOW);
          corrExpectedReturns[j2] = (expectedReturns[j2]/ir)-invTransCost; // changed 20150718: npv = eai/ir + tr. cost // HUGE BUG 20151202: transaction costs should be REDUCED, not added to the npv...
        }

        //int highestReturnFtIndex = getMaxPos(corrExpectedReturns);

        // D - Assigning the Managed area
        // calculating freeArea at the end of the year and choosing the new regeneration area..
        //freeArea(i,essence,lambda) = sum(u, hv2fa(i,essence,lambda,u)*hr(u,i,essence,lambda,t)*V(u,i,lambda,essence,t-1)*100);
        //if(scen("endVreg") ,
        //  regArea(i,essence,lambda,t) = freeArea(i,essence, lambda);   // here we could introduce in/out area from other land usages
        //else
        //  loop (i,
        //    loop( (essence,lambda),
        //      if ( expReturns(i,essence,lambda) = smax( (essence2,lambda2),expReturns(i,essence2,lambda2) ),
        //        regArea (i,essence,lambda,t) =  sum( (essence2, lambda2), freeArea(i,essence2, lambda2) ) * mr;
        //      );
        //    );
        //    regArea(i,essence,lambda,t) = freeArea(i,essence, lambda)*(1-mr);   // here we could introduce in/out area from other land usages
        //  );
        //if (j==highestReturnFtIndex){
        //  thisYearRegAreas[j] += totalHarvestedArea*mr;
        //}
        // If I Implement this I'll have a minimal diff in total area.. why ?????

        double mr = MD->getForData("mr",regId,"","");
        thisYearRegAreas[getMaxPos(corrExpectedReturns)] += harvestedAreaForThisFT*mr;
        thisYearRegAreas[getMaxPos(expectedReturns)] += fArea_incr*mr/((double)fTypes.size()); // mr quota of new forest area assigned to highest expected returns ft (not considering transaction costs). Done for each forest types


        // E - Assigning unmanaged area
        //for(uint j2=0;j2<fTypes.size();j2++){
          if(natRegAllocation=="pp"){ // according to prob presence
            //string ft2 = fTypes[j2];
            string parentFt = MTHREAD->MD->getForTypeParentId(ft);
            double freq = rescaleFrequencies ? gfd("freq_norm",regId,parentFt,""):gfd("freq",regId,parentFt,""); // "probability of presence" for unmanaged forest, added 20140318
            double hAreaThisFtGroup = findMap(hAreaByFTypeGroup,parentFt);
            double hRatio = 1.0;
            if(hAreaThisFtGroup>0){
                //double harvestedAreaForThisFT2 = vSum(px->hArea.at(j2));
                hRatio = harvestedAreaForThisFT/hAreaThisFtGroup;
            } else {
                int nFtChilds = MTHREAD->MD->getNForTypesChilds(parentFt);
                hRatio = 1.0/nFtChilds;
            }
            thisYearRegAreas[j] +=  totalHarvestedArea*(1-mr)*freq*hRatio;
            thisYearRegAreas[j] +=  fArea_incr*(1-mr)*freq*hRatio; // non-managed quota of new forest area assigning proportionally on pp at sp group level
            //thisYearRegAreas[j2] +=  harvestedAreaForThisFT*(1-mr)*freq*hRatio;
          } else { // prob presence not used..

            // Accounting for mortality arising from pathogens. Assigning the area to siblings according to area..


            double mortRatePath = px->getPathMortality(ft, "0");
            if(mortRatePath > 0){

              string parentFt = MTHREAD->MD->getForTypeParentId(ft);
              vector <string> siblings = MTHREAD->MD->getForTypeChilds(parentFt);
              vector <double> siblingAreas;
              for(uint j2=0;j2<siblings.size();j2++){
                if(siblings[j2]==ft){
                  siblingAreas.push_back(0.0);
                } else {
                  string debug_sibling_ft = siblings[j2];
                  int debug_positin = getPos(debug_sibling_ft,fTypes);
                  double thisSiblingArea = vSum(px->area.at(getPos(siblings[j2],fTypes)));
                  siblingAreas.push_back(thisSiblingArea);
                }
              }
              double areaAllSiblings = vSum(siblingAreas);
              thisYearRegAreas[j] += harvestedAreaForThisFT*(1-mr)*(1-mortRatePath);

              if(areaAllSiblings>0.0){ // area of siblings is >0: we attribute the area from the pathogen induced mortality to the siblings proportionally to area..
                for(uint j2=0;j2<siblings.size();j2++){
//                  int debug1 =  getPos(siblings[j2],fTypes);
//                  double debug2= harvestedAreaForThisFT;
//                  double debug3 = 1.0-mr;
//                  double debug4 = mortRatePath;
//                  double debug5 = siblingAreas[j2];
//                  double debug6 = areaAllSiblings;
//                  double debug7 = harvestedAreaForThisFT*(1.0-mr)*(mortRatePath)*(siblingAreas[j2]/areaAllSiblings);
                  thisYearRegAreas[getPos(siblings[j2],fTypes)] += harvestedAreaForThisFT*(1.0-mr)*(mortRatePath)*(siblingAreas[j2]/areaAllSiblings);
                }
              } else if (siblings.size()>1) { // area of all siblings is 0, we just give them the mortality area in equal parts..
                for(uint j2=0;j2<siblings.size();j2++){
                  if (siblings[j2] != ft){
                    thisYearRegAreas[getPos(siblings[j2],fTypes)] += harvestedAreaForThisFT*(1.0-mr)*(mortRatePath)* 1.0 / (( (float) siblings.size())-1.0);
                  }
                }
              }
            } else { // mortRatePath == 0
               thisYearRegAreas[j] += harvestedAreaForThisFT*(1.0-mr);
            }

            // Allocating non-managed quota of new forest area to ft proportionally to the current area share by ft
            double newAreaThisFt = vSum(px->area) ? fArea_incr*(1-mr)*vSum(px->area.at(j))/vSum(px->area): 0.0;
            thisYearRegAreas[j] +=  newAreaThisFt;
            if(! (thisYearRegAreas[j] >= 0.0) ){
              msgOut(MSG_ERROR,"thisYearRegAreas[j] is not non negative (j: "+i2s(j)+", thisYearRegAreas[j]: "+i2s( thisYearRegAreas[j])+").");
            }
            //thisYearRegAreas[j2] += harvestedAreaForThisFT*(1-mr);
          }
        //}
      } // end for each forest type

      // adding regeneration area to the fist (00) diameter class
      for(uint j=0;j<fTypes.size();j++){
        px->area.at(j).at(0) += thisYearRegAreas.at(j);
      }

      #ifdef QT_DEBUG
      double totalRegArea = vSum(thisYearRegAreas);
      if (! (totalRegArea==0.0 && totalHarvestedArea==0.0)){
        double ratio = totalRegArea / totalHarvestedArea ;
        if(rescaleFrequencies && (ratio < 0.99999999999 || ratio > 1.00000000001) ) {
          msgOut(MSG_CRITICAL_ERROR, "Sum of regeneration areas not equal to sum of harvested area in runManagementModel()!");
        }
      }
      #endif
      px->regArea.insert(pair <int, vector<double> > (MTHREAD->SCD->getYear(), thisYearRegAreas));
    } // end of each pixel
    if (-fArea_diff > regHArea){
      msgOut(MSG_WARNING,"In region "+ i2s(regId) + " the exogenous area decrement ("+d2s(-fArea_diff) +" ha) is higger than the harvesting ("+ d2s(regHArea) +" ha). Ratio forced to 1.");
    }

  } // end of each region
}

void
ModelCoreSpatial::cacheSettings(){
  msgOut(MSG_INFO, "Cashing initial model settings..");
  MD = MTHREAD->MD;
  firstYear = MD->getIntSetting("initialYear");
  secondYear = firstYear+1;
  thirdYear  = firstYear+2;
  WL2 = MD->getIntSetting("worldCodeLev2");
  regIds2 = MD->getRegionIds(2);
  priProducts = MD->getStringVectorSetting("priProducts");
  secProducts = MD->getStringVectorSetting("secProducts");
  allProducts = priProducts;
  allProducts.insert( allProducts.end(), secProducts.begin(), secProducts.end() );
  dClasses = MD->getStringVectorSetting("dClasses");
  pDClasses; // production diameter classes: exclude the fist diameter class below 15 cm
  pDClasses.insert(pDClasses.end(), dClasses.begin()+1, dClasses.end() );
  fTypes= MD->getForTypeIds();
  l2r = MD->getRegionIds();
  regType = MTHREAD->MD->getStringSetting("regType"); // how the regeneration should be computed (exogenous, from hr, from allocation choises)
  natRegAllocation = MTHREAD->MD->getStringSetting("natRegAllocation"); // how to allocate natural regeneration
  rescaleFrequencies = MD->getBoolSetting("rescaleFrequencies");
  oldVol2AreaMethod = MD->getBoolSetting("oldVol2AreaMethod");
  //mr = MD->getDoubleSetting("mr");
  forestAreaChangeMethod =  MTHREAD->MD->getStringSetting("forestAreaChangeMethod");
  ir = MD->getDoubleSetting("ir");


}

void
ModelCoreSpatial::initializePixelVolumes(){
  msgOut(MSG_INFO, "Starting initializing pixel-level values");

  // pxVol = regVol * pxArea/regForArea
  // this function can be done only at the beginning of the model, as it assume that the distribution of volumes by diameter class in the pixels within a certain region is homogeneous, but as the model progress along the time dimension this is no longer true.
  if(!MD->getBoolSetting("usePixelData")) return;
  for(uint i=0;i<regIds2.size();i++){
    ModelRegion* reg = MD->getRegion(regIds2[i]);
    vector <Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(
regIds2[i]);
    for (uint j=0;j<rpx.size();j++){
      int debugPx = rpx[j]->getID();
      int debug2 = debugPx;
      rpx[j]->vol.clear(); // not actually necessary
      for(uint y=0;y<fTypes.size();y++){
        vector <double> vol_byu;
        double regForArea = reg->getValue("forArea_"+fTypes[y]);
        for (uint z=0;z<dClasses.size();z++){
          double regVol;
          regVol = z ? gfd("vol",regIds2[i],fTypes[y],dClasses[z],
firstYear) : 0 ; // if z=0-> regVol= gfd(), otherwise regVol=0;
          double pxArea = rpx[j]->getDoubleValue("forArea_"+fTypes[y], true); // bug solved 20121109.
 get zero for not data
          if (pxArea<0.0){
            msgOut(MSG_CRITICAL_ERROR,"Error in initializePixelVolumes, negative
 pxArea!");
          }
          double pxVol = regForArea ? regVol * pxArea/regForArea: 0; // if we introduce new forest types
 without initial area we must avoid a 0/0 division
          //rpx[j]->changeValue(pxVol,"vol",fTypes[y],dClasses[z],firstYear);
          vol_byu.push_back(pxVol);
        }
        rpx[j]->vol.push_back(vol_byu);
      }
    }
  }
  loadExogenousForestLayers("vol");
}

/**
 * @brief ModelCoreSpatial::assignSpMultiplierPropToVols assigns the spatial multiplier (used in the time
 of return) based no more on a Normal distribution but on the volumes present in the pixel: more volume, more
 the pixel is fit for the ft
 *
 * This function apply to the pixel a multiplier of time of passage that is inversely proportional to the
 volumes of that forest type present in the pixel.
 * The idea is that in the spots where we observe more of a given forest type are probabily the most suited
 ones to it.
 *
 * The overall multipliers **of time of passage** (that is, the one returned by
 Pixel::getMultiplier("tp_multiplier") ) will then be the product of this multiplier that account for spatial heterogene
 eventual exogenous
 * multiplier that accounts for different scenarios among the spatio-temporal dimensions.
 *
 * Given that (forest type index omitted):
 * - \f$V_{p}\f$ = volume of a given ft in each pixel (p)
 * - \f$\bar{g}\f$ and \f$\sigma_{g}\f$ = regional average and standard deviation of the growth rate
 * - \f$m_{p}\f$ = multiplier of time of passage
 *
 * This multiplier is computed as:
 * - \f$ v_{p}     = max(V) - V_{p}~~ \f$                    A diff from the max volume is computed in each
 pixel
 * - \f$ vr_{p}    = v_{p} * \bar{g}/\bar{v}~~ \f$           The volume diff is rescaled to match the
 regional growth rate
 * - \f$ vrd_{p}   = vr_{p} - \bar{vr}~~ \f$                 Deviation of the rescaled volumes are computed
 * - \f$ vrdr_{p} = vrd_{p} * \sigma_{g}/\sigma_{vr}~~ \f$ The deviations are then rescaled to match the
 standard deviations of the regional growth rate
 * - \f$ m_{p}     = (vrdr_{p} + \bar{vr}) / \bar{g}~~ \f$  The multiplier is computed from the ratio of the
 average rescaled volumes plus rescaled deviation over the average growth rate.
 *
 * And it has the following properties:
 * - \f$\bar{m} = 1\f$
 * - \f$\sigma_{m} = cv_{g}\f$
 * - \f$m_{p} = V_{p}*\alpha+\beta\f$
 * - \f$m_{\bar{V}} = 1\f$
 *
 * For spreadsheet "proof" see the file
 computation_of_growth_multipliers_from_know_avg_sd_and_proportional_to_share_of_area_in_each_pixel.ods
 */
void
ModelCoreSpatial::assignSpMultiplierPropToVols(){

  if(!MTHREAD->MD->getBoolSetting("useSpatialVarPropToVol")){return;}
  for(uint r=0;r<regIds2.size();r++){
    int rId = regIds2[r];
    ModelRegion* reg = MD->getRegion(regIds2[r]);
    vector <Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(
regIds2[r]);
    for(uint f=0;f<fTypes.size();f++){
      string ft = fTypes[f];
      double agr = gfd("agr",regIds2[r],ft,"");
      double sStDev = gfd("sStDev",regIds2[r],ft,"");
      vector<double> vols;
      vector<double> diffVols;
      vector<double> diffVols_rescaled;
      double diffVols_rescaled_deviation;
      double diffVols_rescaled_deviation_rescaled;
      double final_value;
      double multiplier;
      vector<double> multipliers; // for tests

      double vol_max, rescale_ratio_avg, rescale_ratio_sd;
      double diffVols_avg, diffVols_rescaled_avg;
      double diffVols_rescaled_sd;

      for (uint p=0;p<rpx.size();p++){
        Pixel* px = rpx[p];
        vols.push_back(vSum(px->vol[f]));
      } // end for each pixel
      vol_max=getMax(vols);

      for(uint p=0;p<vols.size();p++){
        diffVols.push_back(vol_max-vols[p]);
      }

      diffVols_avg = getAvg(diffVols);
      rescale_ratio_avg = (diffVols_avg != 0.0) ? agr/diffVols_avg : 1.0;
      for(uint p=0;p<diffVols.size();p++){
        diffVols_rescaled.push_back(diffVols[p]*rescale_ratio_avg);
      }
      diffVols_rescaled_avg = getAvg(diffVols_rescaled);
      diffVols_rescaled_sd  = getSd(diffVols_rescaled,false);

      rescale_ratio_sd = (diffVols_rescaled_sd != 0.0) ? sStdDev/diffVols_rescaled_sd : 1.0;
      for(uint p=0;p<diffVols_rescaled.size();p++){
        diffVols_rescaled_deviation          = diffVols_rescaled[p]     - diffVols_rescaled_avg;
        diffVols_rescaled_deviation_rescaled = diffVols_rescaled_deviation * rescale_ratio_sd;
        final_value                      = diffVols_rescaled_avg    + diffVols_rescaled_deviation_rescaled;
        multiplier = (agr != 0.0) ? min(1.6, max(0.4,final_value/agr)) : 1.0; //20151130: added bounds for extreme cases. Same bonds as in Gis::applySpatialStochasticValues()
       // multiplier = 1.0;

        Pixel* px = rpx[p];
        px->setSpModifier(multiplier,f);
        multipliers.push_back(multiplier);
      }

      #ifdef QT_DEBUG
      // Check relaxed as we introduced bounds that may change slighly the avg and sd...
      double avgMultipliers = getAvg(multipliers);
      double sdMultipliers  = getSd(multipliers,false);
      if ( avgMultipliers < 0.9 || avgMultipliers > 1.1){
        msgOut(MSG_CRITICAL_ERROR, "The average of multipliers of ft "+ ft +" for the region " + i2s(rId) + " is not 1!");
      }
      if ( ( sdMultipliers -  (sStdDev/agr) ) < -0.5 ||  ( sdMultipliers -  (sStdDev/agr) ) > 0.5 ){
        double cv = sStdDev/agr;
        msgOut(MSG_CRITICAL_ERROR, "The sd of multipliers of ft "+ ft +" for the region " + i2s(rId) + " is not equal to the spatial cv for the region!");
      }
      #endif
    } // end for each ft
  } // end for each region
}



void
ModelCoreSpatial::initialiseCarbonModule(){

  ///< call initialiseDeathBiomassStocks(), initialiseProductsStocks() and initialiseEmissionCounters()
  MTHREAD->CBAL->initialiseEmissionCounters();

  for(uint i=0;i<regIds2.size();i++){
    vector<double> deathBiomass;
    for(uint j=0;j<fTypes.size();j++){
      double deathBiomass_ft = gfd("vMort",regIds2[i],fTypes[j], DIAM_ALL,DATA_NOW);
      deathBiomass.push_back(deathBiomass_ft);
    }
    MTHREAD->CBAL->initialiseDeathBiomassStocks(deathBiomass, regIds2[i]);
    vector<double>qProducts;
    for(int p=0;p<priProducts.size();p++){
      // for the primary products we consider only the exports as the domestic consumption is entirelly transformed in secondary products
      double int_exports = gpd("sa",regIds2[i],priProducts[p], DATA_NOW);
      qProducts.push_back(int_exports);
    }
    for(int p=0;p<secProducts.size();p++){
      // for the tranformed product we skip those that are imported, hence derived from other forest systems
      double consumption = gpd("dl",regIds2[i],secProducts[p], DATA_NOW); // dl = sl + net regional imports
      qProducts.push_back(consumption);
    }
    MTHREAD->CBAL->initialiseProductsStocks(qProducts, regIds2[i]);

  }
}

void
ModelCoreSpatial::initialiseDeathTimber(){
  int currentYear = MTHREAD->SCD->getYear();
  for(int y=currentYear;y>currentYear-30;y--){
    for(uint i=0;i<regIds2.size();i++){
      for(uint j=0;j<fTypes.size();j++){
        for (uint u=0;u<dClasses.size();u++){
          iisskey key(y,regIds2[i],fTypes[j],dClasses[u]);
          MD->deathTimberInventory_incrOrAdd(key,0.0);
        }
      }
    }
  }
}

/**
 * @brief ModelCoreSpatial::initializePixelArea
 *
 * This function compute the initial area by ft and dc. It requires vHa computed in computeCumulativeData, this is why it is
 * separated form the other initialisedPixelValues().
 * As the sum of area computed using vHa may differ from the one memorised in forArea_* layer, all values are scaled to match
 * it before being memorised.
 * Also assign area = area_l
 */

void
ModelCoreSpatial::initializePixelArea(){
  msgOut(MSG_INFO, "Starting initializing pixel-level area");
  if(!MD->getBoolSetting("usePixelData")) return;
  for(uint i=0;i<regIds2.size();i++){
    ModelRegion* reg = MD->getRegion(regIds2[i]);
    vector <Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(regIds2[i]);
    for (uint p=0;p<rpx.size();p++){
      Pixel* px = rpx[p];
      double pxid= px->getID();
      for(uint j=0;j<fTypes.size();j++){
        string ft = fTypes[j];
        vector <double> tempAreas;
        vector <double> areasByFt;
        double pxArea = px->getDoubleValue("forArea_"+ft,true)/10000.0; //ha
        for (uint u=0;u<dClasses.size();u++){
          if(u==0){
            double regionArea = reg->getValue("forArea_"+ft,OP_SUM)/10000.0; //ha
            double regRegVolumes = gfd("vReg",regIds2[i],ft,""); // regional regeneration volumes.. ugly name !!
            double newVReg = regionArea ? regRegVolumes*pxArea/regionArea : 0.0;
            double tp_u0 = px->tp.at(j).at(0); // time of passage to reach the first production diameter class
            double entryVolHa = gfd("entryVolHa",regIds2[i],ft,"");
            double tempArea = (newVReg*1000000.0/entryVolHa)*tp_u0;
            tempAreas.push_back(tempArea);
          } else {
            string dc = dClasses[u];
            double dcVol = px->vol_l.at(j).at(u)*1000000.0; // m^3
            double dcVHa = px->vHa.at(j).at(u); // m^3/ha
            #ifdef QT_DEBUG
            if(dcVol < 0.0 || dcVHa < 0.0){
                msgOut(MSG_CRITICAL_ERROR, "Negative volumes or density in initializePixelArea");
            }
            #endif
            double tempArea = dcVHa?dcVol/dcVHa:0;
            tempAreas.push_back(tempArea);
          }

        } // end dc
        double sumTempArea = vSum(tempAreas);
       // double sharedc0 = 5.0/90.0; // an arbitrary share of total area allocated to first diameter class
        //tempAreas.at(0) = sumTempArea * sharedc0;
        //sumTempArea = vSum(tempAreas);
        double normCoef = sumTempArea?pxArea/ sumTempArea:0;
        //cout << i << '\t' << pxid << '\t' << ft << '\t' << normCoef << endl;
        #ifdef QT_DEBUG
        if(normCoef < 0.0){
            msgOut(MSG_CRITICAL_ERROR, "Negative normCoef in initializePixelArea");
        }
        #endif
        for (uint u=0;u<dClasses.size();u++){
          areasByFt.push_back(tempAreas.at(u)*normCoef); //manca la costruzione originale del vettore
        }
        #ifdef QT_DEBUG
        if (pxArea != 0.0){
            double ratio = vSum(areasByFt)/ pxArea;  // vSum(areasByFt) should be equal to pxArea
            if(ratio < 0.9999999999 || ratio > 1.0000000001) {
              msgOut(MSG_CRITICAL_ERROR, "pxArea is not equal to vSum(areasByFt) in initializePixelArea");
            }
        }
        #endif
        px->area_l.push_back(areasByFt);
        /// \todo here I have finally also area_ft_dc_px and I can implement the new one I am in 2006
      } // end ft
      px->area = px->area_l;  //Assigning initial value of area to the area of the old year
    } // end px
  } // end region
  loadExogenousForestLayers("area");
  /// \todo: also update area_l
}

void
ModelCoreSpatial::computeCumulativeData(){

  msgOut(MSG_INFO, "Starting computing some cumulative values..");
  int thisYear     =  MTHREAD->SCD->getYear();

//    double sumCumTP=0;
//    double sumVHa = 0;
//    double count = 0;
//    double avg_sumCumTp;
//    double avg_sumVHa;

  for(uint r2= 0; r2<regIds2.size();r2++){
    int regId = regIds2[r2];
    regPx = MTHREAD->MD->getRegion(regId)->getMyPixels();

    for (uint p=0;p<regPx.size();p++){
      Pixel* px = regPx[p];
      px->cumTp.clear();
      px->cumTp_exp.clear();
      px->vHa_exp.clear();
      px->vHa.clear();
      px->cumAlive.clear();
      px->cumAlive_exp.clear();
      double expType = px->expType;

      for(uint j=0;j<fTypes.size();j++){
        string ft = fTypes[j];

        double tp_multiplier_now       = px->getMultiplier("tp_multiplier",ft, DATA_NOW);
        double tp_multiplier_t0        = px->getMultiplier("tp_multiplier",ft, firstYear);
        double mortCoef_multiplier_now = px->getMultiplier("mortCoef_multiplier",ft, DATA_NOW);
        double mortCoef_multiplier_t0  = px->getMultiplier("mortCoef_multiplier",ft, firstYear);
        double betaCoef_multiplier_now = px->getMultiplier("betaCoef_multiplier",ft, DATA_NOW);
        double betaCoef_multiplier_t0  = px->getMultiplier("betaCoef_multiplier",ft, firstYear);
        double pathMort_now, pathMort_t0;

        // calculating the cumulative time of passage and the (cumulativelly generated) vHa for each diameter class (depending on forest owners diam growth expectations)
        //loop(u$(ord(u)=1),
        //   cumTp(u,i,lambda,essence) = tp_u1(i,essence,lambda);
        //);
        //loop(u$(ord(u)>1),
        //   cumTp(u,i,lambda,essence) = cumTp(u-1,i,lambda,essence)+tp(u-1,i,lambda,essence);
        //);
        ////ceil(x) DNLP returns the smallest integer number greater than or equal to x
        //loop( (u,i,lambda,essence),
        //  cumTp(u,i,lambda,essence) =   ceil(cumTp(u,i,lambda,essence));
        //);
        vector <double> cumTp_temp;     // cumulative time of passage to REACH a diameter class (tp is to LEAVE to the next one)
        vector <double> vHa_temp;       // volume at hectar by each diameter class [m^3/ha]
        vector <double> cumAlive_temp;     // cumulated alive rate to reach a given diameter class
        vector <double> cumTp_exp_temp; // expected version of cumTp_temp
        vector <double> vHa_exp_temp;   // expected version of vHa_temp
        vector <double> cumAlive_exp_temp; // "expected" version of cumMort

        MD->setErrorLevel(MSG_NO_MSG); // as otherwise on 2007 otherwise sfd() will complain that is filling multiple years (2006 and 2007)
        for (uint u=0; u<dClasses.size(); u++){
          string dc = dClasses[u];
          double cumTp_u, cumTp_u_exp, cumTp_u_noExp, cumTp_u_fullExp;
          double tp, tp_exp, tp_noExp, tp_fullExp;
          double vHa_u, vHa_u_exp, vHa_u_noExp, vHa_u_fullExp, beta, beta_exp, beta_noExp, beta_fullExp, mort, mort_exp, mort_noExp, mort_fullExp;
          double cumAlive_u, cumAlive_exp_u;
          pathMort_now = px->getPathMortality(ft,dc,DATA_NOW);
          pathMort_t0 = px->getPathMortality(ft,dc,firstYear);
          // only cumTp is depending for the expectations, as it is what it is used by owner to calculate return of investments.
          // the tp, beta and mort coefficients instead are the "real" ones as predicted by scientist for that specific time

          if(u==0) {
            // first diameter class.. expected  and real values are the same (0)
            cumTp_u = 0.;
            vHa_u   = 0.;
            cumAlive_u = 1.;
            cumTp_temp.push_back(cumTp_u);
            vHa_temp.push_back(vHa_u);
            cumTp_exp_temp.push_back(cumTp_u);
            vHa_exp_temp.push_back(vHa_u);
            cumAlive_temp.push_back(cumAlive_u);
            cumAlive_exp_temp.push_back(cumAlive_u);
          } else {
            // other diameter classes.. first dealing with real values and then with expected ones..
            // real values..
            // real values..
            tp = gfd("tp",regId,ft,dClasses[u-1],thisYear)*tp_multiplier_now;
            cumTp_u = cumTp_temp[u-1] + tp;
            if (u==1){
              /**
              Note on the effect of mortality modifiers on the entryVolHa.
              Unfortunaty for how it is defined the mortality multiplier (the ratio with the new mortality rate over the old one) we can't
              compute a entryVolHa based on it. It is NOT infact just like: vHa_adjusted = vHa_orig / mort_multiplier.
              The effect of mortality on the vHa of the first diameter class is unknow, and so we can't compute the effect of a relative
              increase.
              */
              vHa_u = gfd("entryVolHa",regId,ft,"",thisYear);
              mort = 0.; // not info about mortality first diameter class ("00")
            } else {
              mort = 1-pow(1-gfd("mortCoef",regId,ft,dClasses[u-1],thisYear)*mortCoef_multiplier_now+pathMort_now,tp); // mortality of the previous diameter class
              beta = gfd("betaCoef",regId,ft,dc, thisYear)*betaCoef_multiplier_now;
              vHa_u = vHa_temp[u-1]*beta*(1-mort);
            }
            cumAlive_u = max(0.,cumAlive_temp[u-1]*(1-mort));
            cumAlive_temp.push_back(cumAlive_u);
            cumTp_temp.push_back(cumTp_u);
            vHa_temp.push_back(vHa_u);
            // expected values..
            /**
            param expType Specify how the forest owners (those that make the investments) behave will be the time of passage in the future in order to calculate the cumulative time of passage in turn used to discount future revenues.
            Will forest owners behave adaptively believing the time of passage between diameter classes will be like the observed one at time they make decision (0) or they will have full expectations believing forecasts (1) or something in the middle ?
            For compatibility with the GAMS code, a -1 value means using initial simulation tp values (fixed cumTp)."
            */
            if (expType == -1){
              tp_exp = gfd("tp",regId,ft,dClasses[u-1],firstYear)*tp_multiplier_t0;
              //tp = px->tp.at(u); no. not possible, tp stored at pixel level is the current year one
              cumTp_u_exp = cumTp_exp_temp[u-1]+tp_exp;
              cumTp_exp_temp.push_back(cumTp_u_exp);
              if(u==1) {
                vHa_u_exp = gfd("entryVolHa",regId,ft,"",firstYear);
                mort_exp = 0.; // not info about mortality first diameter class ("00")
              } else {
                mort_exp = 1-pow(1-gfd("mortCoef",regId,ft,dClasses[u-1], firstYear)*mortCoef_multiplier_t0+pathMort_t0,tp_exp); // mortality rate of previous diameter class
                beta_exp = gfd("betaCoef",regId,ft,dc, firstYear)*betaCoef_multiplier_t0;
                vHa_u_exp = vHa_exp_temp[u-1]*beta_exp*(1-mort_exp);
              }
            } else {
              double tp_multiplier_dynamic = px->getMultiplier("tp_multiplier",ft,thisYear+ceil(cumTp_exp_temp[u-1]));
              tp_noExp = gfd("tp",regId,ft,dClasses[u-1])*tp_multiplier_now;
              cumTp_u_noExp = cumTp_exp_temp[u-1]+tp_noExp;
              tp_fullExp = gfd("tp",regId,ft,dClasses[u-1],thisYear+ceil(cumTp_exp_temp[u-1]))*tp_multiplier_dynamic ; // time of passage that there should be to reach this diameter class in the year where the previous diameter class will be reached
              cumTp_u_fullExp = cumTp_exp_temp[u-1]+tp_fullExp ; // it adds to the time of passage to reach the previous diameter class the time of passage that there should be to reach this diameter class in the year where the previous diameter class will be reached
              cumTp_u_exp = cumTp_u_fullExp*expType+cumTp_u_noExp*(1-expType); // 20121108: it's math the same as cumTp_exp_temp[u-1] + tp
              cumTp_exp_temp.push_back(cumTp_u_exp);
              if(u==1) {
                vHa_u_noExp   = gfd("entryVolHa",regId,ft,"",DATA_NOW);
                vHa_u_fullExp = gfd("entryVolHa",regId,ft,"",thisYear+ceil(cumTp_u));
                vHa_u_exp = vHa_u_fullExp*expType+vHa_u_noExp*(1-expType);
                mort_exp = 0.; // not info about mortality first diameter class ("00")
              } else {
                mort_noExp = 1-pow(1-min(1.0,gfd("mortCoef",regId,ft,dClasses[u-1],DATA_NOW)*mortCoef_multiplier_now+pathMort_now), tp_noExp); // mortCoef is a yearly value. Mort coeff between class is 1-(1-mortCoeff)^tp
                double mortCoef_multiplier_dynamic = px->getMultiplier("mortCoef_multiplier",ft,thisYear+ceil(cumTp_exp_temp[u-1]));
                double pathMort_dynamic = px->getPathMortality(ft,dc,thisYear+ceil(cumTp_exp_temp[u-1]));
                mort_fullExp = 1-pow(1-min(1.0,gfd("mortCoef",regId,ft,dClasses[u-1],thisYear+ceil(cumTp_exp_temp[u-1]))*mortCoef_multiplier_dynamic+pathMort_dynamic),tp_fullExp); // mortality of the previous diameter class
                //double debug1 = gfd("mortCoef",regId,ft,dClasses[u-1],thisYear+ceil(cumTp_exp_temp[u-1]));
                //double debug2 = debug1*mortCoef_multiplier_dynamic+pathMort_dynamic;
                //double debug3 = min(1.0,debug2);
                //double debug4 = 1.0-debug3;
                //double debug5 = pow(debug4,tp_fullExp);
                //double debug6 = 1.0-debug5;


                beta_noExp   = gfd("betaCoef",regId,ft,dc, DATA_NOW)*betaCoef_multiplier_now;
                double betaCoef_multiplier_dynamic = px->getMultiplier("betaCoef_multiplier",ft,thisYear+ceil(cumTp_u));
                beta_fullExp = gfd("betaCoef",regId,ft,dc, thisYear+ceil(cumTp_u))*betaCoef_multiplier_dynamic;
                mort_exp = mort_fullExp*expType+mort_noExp*(1-expType);
                beta_exp = beta_fullExp*expType+beta_noExp*(1-expType);
                vHa_u_exp = vHa_exp_temp[u-1]*beta_exp*(1-mort_exp); // BUG !!! mort is yearly value, not between diameter class. SOLVED 20121108
              }
            }
            vHa_exp_temp.push_back(vHa_u_exp);
            cumAlive_exp_u = max(0.,cumAlive_exp_temp[u-1]*(1-mort_exp));
            cumAlive_exp_temp.push_back(cumAlive_exp_u);

            //cout << "********" << endl;
            //cout << "dc;mort;cumAlive;cumAlive_exp "<< endl ;
            //cout << dClasses[u] << ";"<< mort << ";" << cumAlive_u << ";" << cumAlive_exp_u << endl;

          }
          // debug stuff on vHa
          //double vHa_new = gfd("vHa",regId,ft,dc,DATA_NOW);
          //double hv2fa_old = gfd("hv2fa",regId,ft,dc,DATA_NOW);
          //cout << "Reg|Ft|dc|vHa (new)|1/hv2fa (old):   " << regId << " | " << ft;
          //cout << " | " << dc << " | " << vHa_new << " | " << 1/hv2fa_old  << endl;

        } // end of each diam
        //double pixID = px->getID();
        //cout << thisYear << ";"<< regIds2[r2] << ";" << pixID << ";" << ft << ";" << cumTp_exp_temp[3] << ";" << vHa_exp_temp[3] << endl;
        px->cumTp.push_back(cumTp_temp);
        px->vHa.push_back(vHa_temp);
        px->cumAlive.push_back(cumAlive_temp);
        px->cumTp_exp.push_back(cumTp_exp_temp);
        px->vHa_exp.push_back(vHa_exp_temp);
        px->cumAlive_exp.push_back(cumAlive_exp_temp);

        //sumCumTP += cumTp_exp_temp[3];
        //sumVHa += vHa_exp_temp[3];
        //count ++;


      } // end of each ft
      double debug = 0.0;
    } // end of each pixel
  } // end of each region
  MD->setErrorLevel(MSG_ERROR);
    //avg_sumCumTp = sumCumTP/ count;
    //avg_sumVHa = sumVHa / count;
    //cout << "Avg sumCumTp_35 and sumVha_35: " << avg_sumCumTp << " and " << avg_sumVHa << " (" << count << ")" << endl;
    //exit(0);
}

void
ModelCoreSpatial::resetPixelValues(){
  msgOut(MSG_INFO, "Starting resetting pixel level values");
  for(uint r2= 0; r2<regIds2.size();r2++){
    int regId = regIds2[r2];
    regPx = MTHREAD->MD->getRegion(regId)->getMyPixels();
    for (uint p=0;p<regPx.size();p++){
      Pixel* px = regPx[p];
      px->swap(VAR_VOL);  // vol_l = vol
      px->swap(VAR_AREA); // area_l = area
      // 20121108 BUG! Solved, used empty (just return true if the vector is empty) instead of clear (it actually clears the vector)
      px->vol.clear(); // by ft,dc
      px->area = px->area_l; // ATTENTION, DIFFERENT FROM THE OTHERS. Here it is not cleared, it is assigned the previous year as default
      /*px->area.clear(); // by ft,dc*/
      px->hArea.clear(); // by ft, dc
      //px->regArea.clear(); // by year, ft NO, this one is a map, it doesn't need to be changed
      px->hVol.clear(); // by ft, dc
      px->hVol_byPrd.clear(); // by ft, dc, pp
      //px->in.clear(); // by pp
      //px->hr.clear(); // by pp
      px->vReg.clear(); // by ft
      px->expectedReturns.clear(); // by ft

      px->beta.clear();
      px->mort.clear();
      px->tp.clear();
      px->cumTp.clear();
      px->vHa.clear();
      px->cumTp_exp.clear();
      px->vHa_exp.clear();
      px->cumAlive.clear();
      px->cumAlive_exp.clear();
      px->vMort.clear();
      //std::fill(rpx[j]->vMort.begin(), rpx[j]->vMort.end(), 0.0);

    }
  }
}

void
ModelCoreSpatial::cachePixelExogenousData(){

  msgOut(MSG_INFO, "Starting cashing on pixel spatial-level exogenous data");
  for(uint r2= 0; r2<regIds2.size();r2++){
    int regId = regIds2[r2];
    regPx = MTHREAD->MD->getRegion(regId)->getMyPixels();
    for (uint p=0;p<regPx.size();p++){
      Pixel* px = regPx[p];
      px->tp.clear();
      px->beta.clear();
      px->mort.clear();

      for(uint j=0;j<fTypes.size();j++){
        string ft = fTypes[j];
        vector <double> tp_byu;
        vector <double> beta_byu;
        vector <double> mort_byu;

        double tp_multiplier_now       = px->getMultiplier("tp_multiplier",ft, DATA_NOW);
        double mortCoef_multiplier_now = px->getMultiplier("mortCoef_multiplier",ft, DATA_NOW);
        double betaCoef_multiplier_now = px->getMultiplier("betaCoef_multiplier",ft, DATA_NOW);


        for (uint u=0; u<dClasses.size(); u++){
          string dc = dClasses[u];
          double pathMortality           = px->getPathMortality(ft,dc, DATA_NOW);
          double tp, beta_real, mort_real;
          if (u==0){
            // tp of first diameter class not making it change across the time dimension, otherwise problems in getting the rigth past
            // regenerations. BUT good, px->tp.at(0) is used only to pick up the right regeneration, so the remaining of the model
            // uses the getMultiplier version and cumTp consider the dynamic effects also in the first dc.
            tp  = gfd("tp",regId,ft,dClasses[u],firstYear)*px->getMultiplier("tp_multiplier",ft,firstYear); // tp is defined also in the first
diameter class, as it is the years to reach the NEXT diameter class
          } else {
            tp  = gfd("tp",regId,ft,dClasses[u],DATA_NOW)*tp_multiplier_now; // tp is defined also in the first diameter class, as it is the years to reach the NEXT diameter class
          }
          beta_real = u?gfd("betaCoef",regId,ft,dClasses[u],DATA_NOW)*betaCoef_multiplier_now:0;
          mort_real = min(u?gfd("mortCoef",regId,ft,dClasses[u],DATA_NOW)*mortCoef_multiplier_now+pathMortality :0,1.0);  //Antonello, bug fixed 20160203: In any case, natural plus pathogen mortality can not be larger than 1!
          tp_byu.push_back(tp);
          beta_byu.push_back(beta_real);
          mort_byu.push_back(mort_real);
        } // end of each tp
        px->tp.push_back(tp_byu);
        px->beta.push_back(beta_byu);
        px->mort.push_back(mort_byu);
      } // end of each ft
    } // end of each pixel
  } // end of each region
}

void
ModelCoreSpatial::computeInventary(){ // in=f(vol_t-1)
  msgOut(MSG_INFO, "Starting computing inventary available for this year..");
  int nbounds = pow(2,priProducts.size());
  vector<vector<int>> concernedPriProductsTotal = MTHREAD->MD->createCombinationsVector(priProducts.size());
  int currentYear = MTHREAD->SCD->getYear();

  for(uint i=0;i<regIds2.size();i++){
    int r2 = regIds2[i];
    ModelRegion* REG = MTHREAD->MD->getRegion(r2);
    //Gis* GIS = MTHREAD->GIS;
    regPx = REG->getMyPixels();
    vector <double> in_reg(priProducts.size(),0.);             // should have ceated a vector of size priProducts.size(), all filled with zeros
    vector <double> in_deathTimber_reg(priProducts.size(),0.); // should have ceated a vector of size priProducts.size(), all filled with zeros
    for (uint p=0;p<regPx.size();p++){
      Pixel* px = regPx[p];
      //int debugPx = px->getID();
      //int debug2 = debugPx;
      //px->in.clear();
      for(uint pp=0;pp<priProducts.size();pp++){
        double in = 0;
        for(uint ft=0;ft<fTypes.size();ft++){
          for(uint dc=0;dc<dClasses.size();dc++){
            in += app(priProducts[pp],fTypes[ft],dClasses[dc])*px->vol_l.at(ft).at(dc)*px->avalCoef;
          }
        }
        //px->in.push_back(in);
        in_reg.at(pp) += in;
      } // end of each priProduct
    } // end each pixel


    for(uint pp=0;pp<priProducts.size();pp++){
      vector<string> priProducts_vector;
      priProducts_vector.push_back(priProducts[pp]);

      double in_deathMortality = MD->getAvailableDeathTimber(priProducts_vector,r2,currentYear-1);
      in_deathTimber_reg.at(pp) += in_deathMortality;

      // Even if I fixed all the lower bounds to zero in Opt::get_bounds_info still the model
      // doesn't solve with no-forest in a region.
      // Even with 0.0001 doesn't solve !!
      // With 0.001 some scenarios doesn't solve in 2093
      // With 0.003 vRegFixed doesn't solve in 2096
      // Tried with 0.2 but no changes, so put it back on 0.003
      //spd(max(0.001,in_reg.at(pp)),"in",r2,priProducts[pp],DATA_NOW,true);
      spd(in_reg.at(pp),"in",r2,priProducts[pp],DATA_NOW,true);
      spd(in_deathTimber_reg.at(pp),"in_deathTimber",r2,priProducts[pp],DATA_NOW,true);
      #ifdef QT_DEBUG
      if (in_reg.at(pp) < -0.0){
        msgOut(MSG_CRITICAL_ERROR,"Negative inventory");
      }
      #endif
    }

    // ##### Now creating a set of bonds for the optimisation that account of the fact that the same ft,dc can be used for multiple products:

    // 20160928: Solved a big bug: for each combination instead of taking the UNION of the various
```

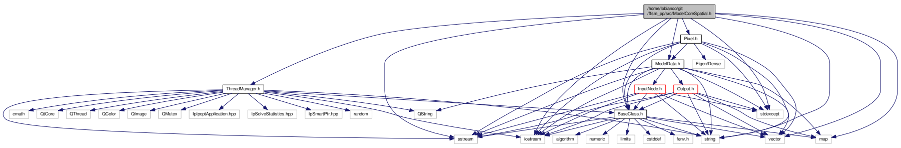
```
priProduct inventory sets I was taking the sum
    // Now both the alive and the death timber are made from the union
    // 20150116: As the same (ft,dc) can be used in more than one product knowing -and bounding the supply in the optimisation- each single
    // in(pp) is NOT enought.
    // We need to bound the supply for each possible combination, that is for 2^(number of prim.pr)
    // Here we compute the detailed inventory. TO.DO: Create the pounds in Opt. done
    // 20160209: Rewritten and corrected a bug that was not giving enought inv to multiproduct combinations
    for (uint i=0; i<nbounds; i++){
      vector<int> concernedPriProducts = concernedPriProductsTotal[i];
      vector<string> concernedPriProducts_ids = positionsToContent(priProducts,concernedPriProducts);
      //double debug       = 0.0;
      //for(uint z=0;z<concernedPriProducts.size();z++){
      //  debug      += gpd("in",r2,priProducts[concernedPriProducts[z]]); // to.do: this will need to be rewritten checked!
      //}
      double bound_alive       = MD->getAvailableAliveTimber(concernedPriProducts_ids,r2); // From px->vol_l, as in "in"
      double bound_deathTimber = MD->getAvailableDeathTimber(concernedPriProducts_ids,r2,currentYear-1); // From deathTimberInventory map
      double bound_total       = bound_alive + bound_deathTimber;

      REG->inResByAnyCombination[i]             = bound_total;
      REG->inResByAnyCombination_deathTimber[i] = bound_deathTimber;
    } // end for each bond
  } // end each region
}

void
ModelCoreSpatial::updateMapAreas(){
  msgOut(MSG_INFO, "Updating map areas..");

  if (!oldVol2AreaMethod){
    if(!MD->getBoolSetting("usePixelData")) return;
    for(uint i=0;i<regIds2.size();i++){
      ModelRegion* reg = MD->getRegion(regIds2[i]);
      vector <Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(regIds2[i]);
      for (uint p=0;p<rpx.size();p++){
        Pixel* px = rpx[p];
        double pxid= px->getID();
        for(uint j=0;j<fTypes.size();j++){
          string ft = fTypes[j];
          double forArea = vSum(px->area.at(j));
          #ifdef QT_DEBUG
          if(forArea < 0.0 ){
            msgOut(MSG_CRITICAL_ERROR, "Negative forArea in updateMapAreas");
          }
          #endif
          px->changeValue("forArea_"+ft, forArea*10000);
        } // end ft
      } // end px
    } // end region
  } else {
    int currentYear = MTHREAD->SCD->getYear();
    map<int,double> forestArea; // foresta area by each region
    pair<int,double > forestAreaPair;
    vector<int> l2Regions =  MTHREAD->MD->getRegionIds(2, true);
    vector <string> fTypes =  MTHREAD->MD->getForTypeIds();
    int nFTypes = fTypes.size();
    int nL2Regions = l2Regions.size();
    for(int i=0;i<nL2Regions;i++){
      int regId = l2Regions[i];
      vector<Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(regId);
      for(int j=0;j<nFTypes;j++){
        string ft = fTypes[j];
        //double regForArea = reg->getValue("forArea_"+ft);
        //double harvestedArea = gfd("harvestedArea",regId,ft,DIAM_ALL);
        //double regArea = gfd("regArea",regId,ft,DIAM_ALL);
        //cout << "Regid/ft/area/harvested/regeneration: " <<regId<<";"<<ft<<";"<<regForArea<<";"<<harvestedArea<<";" <<regArea<<endl;
        //double newAreaNet = regArea-harvestedArea;
        //double newAreaRatio =  newAreaNet / regForArea;
        for(uint z=0;z<rpx.size();z++){
          Pixel* px = rpx[z];
          double oldValue = px->getDoubleValue("forArea_"+ft,true)/10000;
          double hArea = vSum(px->hArea.at(j));              //bug 20140205 areas in the model are in ha, in the layer in m^2
          double regArea = findMap(px->regArea,currentYear).at(j); //bug 20140205 areas in the model are in ha, in the layer in m^2
          //double newValue = oldValue*(1. + newAreaRatio);
          double newValue = oldValue-hArea+regArea;
          double areaNetOfRegeneration = oldValue-hArea;
          #ifdef QT_DEBUG
          if (areaNetOfRegeneration<0.0){
            msgOut(MSG_CRITICAL_ERROR,"areaNetOfRegeneration negative in
 updateMapAreas");
          }
          if (newValue<0.0){
            msgOut(MSG_CRITICAL_ERROR,"for area negative in updateMapAreas");
          }
          #endif
          rpx[z]->changeValue("forArea_"+ft, newValue*10000);
        }
      }
    }
  }
}

void
ModelCoreSpatial::updateOtherMapData(){

vector<int> l2Regions =  MTHREAD->MD->getRegionIds(2, true);
vector <string> fTypes =  MTHREAD->MD->getForTypeIds();
int nFTypes = fTypes.size();
int nL2Regions = l2Regions.size();
for(int i=0;i<nL2Regions;i++){
  int regId = l2Regions[i];
  vector<Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(regId);
  for(int j=0;j<nFTypes;j++){
    string ft = fTypes[j];
    for(uint z=0;z<rpx.size();z++){
      Pixel* px = rpx[z];
      double vol = vSum(px->vol.at(j));
      double expectedReturns = px->expectedReturns.at(j);
      if(MTHREAD->GIS->layerExist("vol_"+ft)){
        rpx[z]->changeValue("vol_"+ft, vol);
      }
      if(MTHREAD->GIS->layerExist("expectedReturns_"+ft)){
        rpx[z]->changeValue("expectedReturns_"+ft, expectedReturns);
      }
    }
  }
}

// update GUI image..
for(int j=0;j<nFTypes;j++){
  string ft = fTypes[j];
  MTHREAD->GIS->updateImage("vol_"+ft);
  MTHREAD->GIS->updateImage("expectedReturns_"+ft);
}


}


void
ModelCoreSpatial::sumRegionalForData(){

  msgOut(MSG_INFO, "Summing data pixels->region..");
  //vector <string> outForVariables  = MTHREAD->MD->getStringVectorSetting("outForVariables");
  int currentYear = MTHREAD->SCD->getYear();

  // OLD CODE TO
  for(uint r2= 0; r2<regIds2.size();r2++){
    int regId = regIds2[r2];
    regPx = MTHREAD->MD->getRegion(regId)->getMyPixels();

    for(uint j=0;j<fTypes.size();j++){
      string ft = fTypes[j];

      double regArea = 0.;
      double sumAreaByFt = 0.;
      double pxForAreaByFt = 0.;
      double vReg = 0.;

      for (uint u=0; u<dClasses.size(); u++){
        string dc = dClasses[u];
        double vol =0.;
        double hV = 0.;
        double hArea = 0.;
        double vMort = 0.;
        for (uint p=0;p<regPx.size();p++){
          Pixel* px = regPx[p];
          vol += px->vol.at(j).at(u);
          hV += px->hVol.at(j).at(u);
          hArea += px->hArea.at(j).at(u);
          vMort += px->vMort.at(j).at(u);
        }
        if(u){
          sfd(vol,"vol",regId,ft,dc,DATA_NOW);
          sfd(hV,"hV",regId,ft,dc,DATA_NOW,true);
          sfd(hArea,"harvestedArea",regId,ft,dc,DATA_NOW, true);
          sfd(vMort,"vMort",regId,ft,dc,DATA_NOW,true);
          double vol_1 = gfd("vol",regId,ft,dc,currentYear-1);
          if(vol_1){
            sfd(hV/vol_1,"hr",regId,ft,dc,DATA_NOW, true);
          } else {
            sfd(0.,"hr",regId,ft,dc,DATA_NOW, true);
          }

        }
      }
      for (uint p=0;p<regPx.size();p++){
        Pixel* px = regPx[p];
        vReg += px->vReg.at(j);
        regArea += findMap(px->regArea,currentYear).at(j);
        pxForAreaByFt = (px->getDoubleValue("forArea_"+ft,true)/10000);

        sumAreaByFt += pxForAreaByFt;
        //double debug1 = sumAreaByFt;
        if(! (sumAreaByFt >= 0.0) ){
          msgOut(MSG_CRITICAL_ERROR,"sumAreaByFt is not non negative.");
        }
      }
      sfd(vReg,"vReg",regId,ft,"",DATA_NOW, true);
      sfd(regArea,"regArea",regId,ft,"",DATA_NOW, true);
      sfd(sumAreaByFt,"forArea",regId,ft,"",DATA_NOW, true);
    } // end of for each ft

    for (uint p=0;p<regPx.size();p++){
      Pixel* px = regPx[p];
      double totPxForArea = vSum(px->area);

#ifdef QT_DEBUG
      double totPxForArea_debug = 0.0;
      for(uint j=0;j<fTypes.size();j++){
        string ft = fTypes[j];
        totPxForArea_debug += (px->getDoubleValue("forArea_"+ft,true)/10000);
      }

      if ( (totPxForArea - totPxForArea_debug) > 0.0001 || (totPxForArea - totPxForArea_debug) < -0.0001 ){
        cout << "*** ERROR: area discrepance in pixel " << px->getID() << " of " << (totPxForArea - totPxForArea_debug) << " ha!" << endl;
        msgOut(MSG_CRITICAL_ERROR,"Total forest area in pixel do not coincide if token from layer forArea or (pixel) vector area!");
      }
#endif
    } // end of each pixel

  } // end each region



  // Taking care of expected returns here..
  // (Changed 25/08/2016 afternoon: expRet{ft,r} are now sum{px}{expRet{ft,px}*fArea_{px}}/fArea{r} and no longer sum{px}{expRet{ft,px}*fArea_{px,ft}}/fArea{r,ft} )
  // Also now we report the expReturns by group and by forest, each of which is made only with the best ones within their group

  vector<string>  parentFtypes = MTHREAD->MD->getForTypeParents();

  for(uint r2= 0; r2<regIds2.size();r2++){
    int regId = regIds2[r2];
    regPx = MTHREAD->MD->getRegion(regId)->getMyPixels();
    double totRegionForArea = 0.;
    double totSumExpRet  = 0.;
    vector <double> totSumExpRet_byFTParent(parentFtypes.size(),0.0);
    vector <double> totSumExpRet_byFTypes(fTypes.size(),0.0);

    // First computing the sumExpectedReturns..
    for (uint p=0;p<regPx.size();p++){
      Pixel* px = regPx[p];
      //int debug_pxid = px->getID();
      double pxForArea = vSum(px->area);
      totRegionForArea += pxForArea;
      double bestPxExpectedRet = getMax(px->expectedReturnsNotCorrByRa);
      for(uint i=0;i<parentFtypes.size();i++){
        vector <string> childIds = MTHREAD->MD->getForTypeChilds(parentFtypes[i]);
        vector <int> childPos = MTHREAD->MD->getForTypeChilds_pos(parentFtypes[i]);
        vector<double> pxExpReturnsByChilds(childPos.size(),0.0);
        for(uint j=0;j<childPos.size();j++){
          double pxExpReturn_singleFt = px->expectedReturns.at(childPos[j]);
          // Manual fix to not have the expected returns of ash within the general "broadL" expected returns.
          // To do: remove it after we work on the ash project.. I don't like manual fixes !!!
          pxExpReturnsByChilds.at(j) = (childIds.at(j) == "ash") ? 0.0 : pxExpReturn_singleFt;
          //pxExpReturnsByChilds.at(j) = pxExpReturn_singleFt;
          totSumExpRet_byFTypes.at(childPos[j]) += pxExpReturn_singleFt*pxForArea;
        } // end of each ft
        totSumExpRet_byFTParent[i] += getMax(pxExpReturnsByChilds)*pxForArea;
      } // end for each partentFt
      totSumExpRet += bestPxExpectedRet * pxForArea;
    } // end for each px

    // ..and now computing the expReturns and storing them
    for(uint i=0;i<parentFtypes.size();i++){
      vector <int> childPos = MTHREAD->MD->getForTypeChilds_pos(parentFtypes[i]);
      for(uint j=0;j<childPos.size();j++){
        //double debug1 = totSumExpRet_byFTypes.at(childPos[j])/totRegionForArea;
        sfd(totSumExpRet_byFTypes.at(childPos[j]),"sumExpReturns",regId,fTypes.at(childPos[j]),"",DATA_NOW, true);
        sfd(totSumExpRet_byFTypes.at(childPos[j])/totRegionForArea,"expReturns",regId,fTypes.at(childPos[j]),"",DATA_NOW, true);
      } // end of each ft
      //double debug2 = totSumExpRet_byFTParent.at(i)/totRegionForArea;
      sfd(totSumExpRet_byFTParent.at(i),"sumExpReturns",regId,parentFtypes[i],"",DATA_NOW, true);
      sfd(totSumExpRet_byFTParent.at(i)/totRegionForArea,"expReturns",regId,parentFtypes[i],"",DATA_NOW, true);

    } // end for each partentFt
    //double debug3 = totSumExpRet/totRegionForArea;
    sfd(totSumExpRet,"sumExpReturns",regId,"","",DATA_NOW, true);
    sfd(totSumExpRet/totRegionForArea,"expReturns",regId,"","",DATA_NOW, true);

  } // end for each region

  // Computing pathogens share of forest invasion
  if(MD->getBoolSetting("usePathogenModule")){
      for(uint r2= 0; r2<regIds2.size();r2++){
        int regId = regIds2[r2];
        regPx = MTHREAD->MD->getRegion(regId)->getMyPixels();
        double totalForArea = 0.0;
        double invadedArea  = 0.0;
        for (uint p=0;p<regPx.size();p++){
          Pixel* px = regPx[p];
          int invaded = 0.0;
          for(uint j=0;j<fTypes.size();j++){
            for (uint u=0; u<dClasses.size(); u++){
              if(px->getPathMortality(fTypes[j],dClasses[u]) > 0){
                invaded = 1.0;
              }
            }
          }
          totalForArea += vSum(px->area);
          invadedArea  += vSum(px->area)*invaded;
        }
        sfd(invadedArea/totalForArea,"totalShareInvadedArea",regId,"","",DATA_NOW, true);
      }
  } // end we are using path model
}
/**
 * This function call registerHarvesting() (accounts for emissions from for. operations), registerDeathBiomass() (registers new stocks of death biomass),
 * registerProducts() (registers new stock of products) and registerTransports() (accounts for emissions from transportation).
 *
 * It pass to registerProducts():
 * - for primary products, the primary products exported out of the country, but not those exported to other regions or used in the region as
 *   these are assumed to be totally transformed to secondary products;
 * - for secondary products, those produced in the region from locally or regionally imported primary product plus those secondary products
 *   imported from other regions, less those exported to other regions. It doesn't include the secondary products imported from abroad the country.
 */
void
ModelCoreSpatial::registerCarbonEvents(){

  //void                  registerHarvesting(const int & regId, const string & fType, const double & value); ///< register the harvesting of trees -> cumEmittedForOper
  //void                  registerDeathBiomass(const double &value, const int & regId, const string &fType);
  //void                  registerProducts(const double &value, const int & regId, const string &productName);
  //void                  registerTransports(const double &distQ, const int & regId);

  for(uint i=0;i<regIds2.size();i++){
    for(uint j=0;j<fTypes.size();j++){
      double deathBiomass = gfd("vMort",regIds2[i],fTypes[j],DIAM_ALL,DATA_NOW);
      double harvesting = gfd("hV",regIds2[i],fTypes[j],DIAM_ALL,
DATA_NOW);
      MTHREAD->CBAL->registerDeathBiomass(deathBiomass, regIds2[i], fTypes[j]);  // register new stock
      MTHREAD->CBAL->registerHarvesting(harvesting, regIds2[i], fTypes[j]);      // account for emissions. Added 201500715: it also moves the extra biomass to the death biomass pool
    }

    for(uint p=0;p<priProducts.size();p++){
      // for the primary products we consider only the exports as the domestic consumption is entirelly transformed in secondary products
      double int_exports = gpd("sa",regIds2[i],priProducts[p], DATA_NOW);
      MTHREAD->CBAL->registerProducts(int_exports, regIds2[i], priProducts[p]);  // register new stock
    }
    for(uint p=0;p<secProducts.size();p++){
      // for the tranformed product we skip those that are imported, hence derived from other forest systems
      // but we consider those coming from other regions
      double consumption = gpd("dl",regIds2[i],secProducts[p], DATA_NOW); // dl = sl + net regional imports
      MTHREAD->CBAL->registerProducts(consumption, regIds2[i], secProducts[p]); // register new stock
    }

  }
  for (uint r1=0;r1<l2r.size();r1++){
    for (uint r2=0;r2<l2r[r1].size();r2++){
      int rfrom= l2r[r1][r2];
      double distQProd = 0.0;
      for (uint r3=0;r3<l2r[r1].size();r3++){
        int rto = l2r[r1][r3];
        double dist = gpd("dist",rfrom,"",DATA_NOW,i2s(rto)); //km
        for(uint p=0;p<allProducts.size();p++){
          distQProd += dist*gpd("rt",rfrom,allProducts[p],DATA_NOW, i2s(rto)); //km*Mm^3
        }
      }
      MTHREAD->CBAL->registerTransports(distQProd, rfrom);
    }
  }
  MTHREAD->CBAL->HWP_eol2energy(); // used to compute the energy substitution from hwp that reach the end of life and doesn't go to landfil. Previously the energy substitution was computed in registerProducts(), that is at the time when the product was produced.

}

/**
 * Compute the expectation weighted price based on the ratio of the international (world) price between the future and now.
 *
 * @param curLocPrice The local current price
 * @param worldCurPrice The world current price
 * @param worldFutPrice The world future price
 * @param sl Supply local
 * @param sa Supply abroad
 * @param expCoef The expectation coefficient for prices for the agent [0,1]
 * @return The expType-averaged local (or weighter) price
 */
double
ModelCoreSpatial::computeExpectedPrice(const double & curLocPrice, const double & worldCurPrice, const double & worldFutPrice, const double & sl, const double & sa, const double & expCoef){
  double fullExpWPrice = (curLocPrice*(worldFutPrice/worldCurPrice)*sl+worldFutPrice*sa)/(sa+sl);
  double curWPrice = (curLocPrice*sl+worldCurPrice*sa)/(sl+sa);
  return curWPrice * (1-expCoef) + fullExpWPrice * expCoef;
}

/**
 * It uses volumes from gis data to "move" volumes from one forest type to the other (when called with what="vol"). Then it moves areas
 * proportionally and, as dc0 volumes are not defined but area it is, compute, again proportionally, area in destination forest times for dc=0
 * It acts on the pix->vol, pix->area and pix->area_l vectors. It also create/update the px->values layer map for the area, but it doesn't cash the
 * results in forDataMap.
 *
 * It is called first with parameter what="vol" in initializePixelVolumes() and then with what="area" in initializePixelAreas().
 * As we need the original volumes in the area allocation, original_vols is set as a static variable.
 *
 */
void
ModelCoreSpatial::loadExogenousForestLayers(const string & what) {
  if(!MD->getBoolSetting("useSpExplicitForestTypes")) return;

  int nFTypes = fTypes.size();
  int nDC     = dClasses.size();
  int pxC     = 0;

  for(uint ir=0;ir<regIds2.size();ir++){
    int r2 = regIds2[ir];
    ModelRegion* REG = MTHREAD->MD->getRegion(r2);
    regPx = REG->getMyPixels();
    pxC += regPx.size();
  }

  static vector<vector<vector<double>>> original_vols(pxC, vector<vector<double>>(nFTypes, vector<double>(nDC, 0.0))); // by px counter, ftype, dc

  if(what=="vol"){
    // first, before transfering volumes, saving the original ones..
    for(uint i=0;i<fTypes.size();i++){
      for (uint u=0; u<dClasses.size(); u++){
        int pxC_loc = 0;
        for(uint ir=0;ir<regIds2.size();ir++){
          int r2 = regIds2[ir];
          ModelRegion* REG = MTHREAD->MD->getRegion(r2);
          regPx = REG->getMyPixels();
          for (uint p=0;p<regPx.size();p++){
            Pixel* px = regPx[p];
            original_vols[pxC_loc][i][u] += px->vol[i][u];
            pxC_loc ++;
          }
        }
      }
    }
    for(uint i=0;i<fTypes.size();i++){
      string fti = fTypes[i];
      for(uint o=0;o<fTypes.size();o++){
        string fto = fTypes[o];
        for (uint u=1; u<dClasses.size(); u++){ // first diameter class volumes are computed from the model..
          string layerName =  "spInput#vol#"+fto+"#"+fti+"#"+i2s(u);
          if (MTHREAD->GIS->layerExist(layerName)){
            for(uint ir=0;ir<regIds2.size();ir++){
              int r2 = regIds2[ir];
              ModelRegion* REG = MTHREAD->MD->getRegion(r2);
              regPx = REG->getMyPixels();
              for (uint p=0;p<regPx.size();p++){
                Pixel* px = regPx[p];
                double vol_transfer = min(px->getDoubleValue(layerName,true)/1000000,px->vol[i][u]) ; // Vol in the layer are in m^3, in the model in Mm^3
                px->vol[i][u] -= vol_transfer;
                px->vol[o][u] += vol_transfer;
              }
            }
          }
        }
      }
    }
  }

  if(what=="area"){
    /**
    Allocate area proportionally to volumes (see file test_proportional_computation_of_areas_from_volumes.ods)
    Example:
    FtIn   FtOut Vtrasfer
    con    ash   0.2
    brHf   ash   0.1
    brCopp ash   0.3
    con    oak   0.3
    brHf   oak   0.2
    brCopp oak   0.1

           Vorig Aorig Vnew  Anew
    con    10    30    9.5   28.5  Aorig-Aorig*(Vtrasfer1/Vorig)-Aorig(Vtrasfer2/Vorig)
    brHf   5     20    4.7   18.8
    brCopp 2     20    1.6   16
    ash    0     0     0.6   4     Aorig1*Vtrasfer1/(Vorig1)+Aorig2*Vtrasfer2/(Vorig2)+...
    oak    0     0     0.6   2.7
                 70          70
    */
    // first, before transfering areas, saving the original ones (we already saved the vols in the what="vol" section, that is called before this one)..
    vector<vector<vector<double>>> original_areas(pxC, vector<vector<double>>(nFTypes, vector<double>(nDC, 0.0))); // by px counter, ftype, dc
    for(uint i=0;i<fTypes.size();i++){
      for (uint u=0; u<dClasses.size(); u++){
        int pxC_loc = 0;
        for(uint ir=0;ir<regIds2.size();ir++){
          int r2 = regIds2[ir];
          ModelRegion* REG = MTHREAD->MD->getRegion(r2);
          regPx = REG->getMyPixels();
          for (uint p=0;p<regPx.size();p++){
            Pixel* px = regPx[p];
            original_areas[pxC_loc][i][u] += px->area_l[i][u];
            pxC_loc ++;
          }
        }
      }
    }


    // transferred areas ordered by pxcounter, i and then o ftype. Used to then repart the 0 diameter class..
    vector<vector<vector<double>>> transferred_areas(pxC, vector<vector<double>>(nFTypes, vector<double>(nFTypes, 0.0))); // initialize a 3d vector of nFTypes zeros.

    for(uint i=0;i<fTypes.size();i++){
      string fti = fTypes[i];
      for(uint o=0;o<fTypes.size();o++){
        string fto = fTypes[o];
        for (uint u=1; u<dClasses.size(); u++){ // first diameter class area is comuted proportionally..
          string layerName =  "spInput#vol#"+fto+"#"+fti+"#"+i2s(u);
          if (MTHREAD->GIS->layerExist(layerName)){
            int pxC_loc = 0;
            for(uint ir=0;ir<regIds2.size();ir++){
              int r2 = regIds2[ir];
              ModelRegion* REG = MTHREAD->MD->getRegion(r2);
              regPx = REG->getMyPixels();
              for (uint p=0;p<regPx.size();p++){
                Pixel* px = regPx[p];
                double vol_i_orig    = original_vols[pxC_loc][i][u];
                double vol_transfer  = vol_i_orig?px->getDoubleValue(layerName,true)/1000000:0.0; // Vol in the layer are in m^3, in the model in Mm^3
                double area_i_orig   = original_areas[pxC_loc][i][u];
                double area_transfer = vol_i_orig?area_i_orig*vol_transfer/vol_i_orig:0.0;
                px->area_l[i][u] -=  area_transfer;
                px->area[i][u]    = px->area_l[i][u];
                px->area_l[o][u] += area_transfer;
                px->area[o][u]    = px->area_l[o][u];
                transferred_areas[pxC_loc][i][o] += area_transfer;
                pxC_loc ++;
             }
           }
          }
        }
      }
    }

    // Moving the area in the 0 diameter class, for which no info is normally available..
    double pxC_loc = 0;
    for(uint ir=0;ir<regIds2.size();ir++){
      int r2 = regIds2[ir];
      ModelRegion* REG = MTHREAD->MD->getRegion(r2);
      regPx = REG->getMyPixels();
      for (uint p=0;p<regPx.size();p++){
        Pixel* px = regPx[p];
        for(uint i=0;i<fTypes.size();i++){
          for(uint o=0;o<fTypes.size();o++){
            double area_i_orig = 0.0;
            for (uint u=1; u<dClasses.size(); u++){ // we want to skip the 0 diameter class, this is why we don't simply use vSum()..
              area_i_orig += original_areas[pxC_loc][i][u];
            }
            double area_transfer_u0 = area_i_orig?original_areas[pxC_loc][i][0]*(transferred_areas[pxC_loc][i][o]/area_i_orig):0.0;
            px->area_l[i][0] -= area_transfer_u0 ;
            px->area[i][0] = px->area_l[i][0];
            px->area_l[o][0] += area_transfer_u0 ; // bug corrected 20151130: it was 0 instead of o (output) !!
            px->area[o][0] = px->area_l[0][0];     // bug corrected 20151130: it was 0 instead of o (output) !!
          }
        }
        pxC_loc++;
      }
    }

    // Alligning the area memorised in the px layers to the new areas of the ft..
    for(uint i=0;i<fTypes.size();i++){
      string fti_id = fTypes[i];
      forType* fti  = MTHREAD->MD->getForType(fti_id);
      int ft_memType = fti->memType;
      string ft_layerName = fti->forLayer;
      //if(ft_memType==3){
```

```
      //  MTHREAD->GIS->addLayer(ft_layerName,ft_layerName,false,true); //20151130: no needed as we already added it in applyForestReclassification (yes, odd, as memory type 3 layerrs do not have any reclassification rule associated, but if I don't add the layer at that time I got other errors)
      // }
      if(ft_memType==3 ||ft_memType==2){
        for(uint ir=0;ir<regIds2.size();ir++){
          int r2 = regIds2[ir];
          ModelRegion* REG = MTHREAD->MD->getRegion(r2);
          regPx = REG->getMyPixels();
          for (uint p=0;p<regPx.size();p++){
            Pixel* px = regPx[p];
            double area_px = vSum(px->area[i]);
            px->changeValue(ft_layerName,area_px*10000);
          }
        }
      }
    }
  } // end if what is area
}

void
ModelCoreSpatial::printDebugInitRegionalValues(){
  // Print debug stats on inventory and supplies in each region..
  cout << "Printing debug information on initial regional inventories and supplies.." << endl;
  cout << "Reg\tProduct\t\tInv\tSt\tSa\tSl" << endl;
  for(uint r1=0;r1<l2r.size();r1++){
    for(uint r2c=0;r2c<l2r[r1].size();r2c++){
      for(uint p=0;p<priProducts.size();p++){
        int r2 = l2r[r1][r2c];
        double inv = gpd("in",r2,priProducts[p],secondYear);
        double st = gpd("st",r2,priProducts[p],secondYear);
        double sl = gpd("sl",r2,priProducts[p],secondYear);
        double sa = gpd("sa",r2,priProducts[p],secondYear);
        cout << r2 << "\t" << priProducts[p] << "\t\t" << inv << "\t" << st << "\t" << sl << "\t" << sa << endl;
      }
    }
  } // end of r1 region
  exit(0);

}

/**
 * @brief ModelCoreSpatial::allocateHarvesting
 * @param total_st vector of total supply by primary products
 * @return a vector of the remaining supply that goes allocated to alive timber (that is, to harvesting)
 *
 * The algorithm is such that is loops the deathTimberInventory map for each year (newer to older), dc (higher to smaller) and ft.
 * It compute the primary products allocable from that combination and allocate the cell amount to decrease the total_st of that products
 * in a proportional way to what still remain of the allocable products.
 *
 * It is called in the runMarketModule() function.
 *
 */

vector <double>
ModelCoreSpatial::allocateHarvesting(vector<double> total_st, const int &regId){
   if(!MD->getBoolSetting("useDeathTimber")) return total_st;
   vector <double> stFromHarvesting(priProducts.size(),0.);
   //map<iisskey, double > *  deathTimberInventory= MD->getDeathTimberInventory();
   int maxYears = MD->getMaxYearUsableDeathTimber();
   int currentYear = MTHREAD->SCD->getYear();
   for(uint y = currentYear-1; y>currentYear-1-maxYears; y--){
     for (int u = dClasses.size()-1; u>=0; u--){  // I need to specify u as an integer !
       string dc = dClasses.at(u);
       for (uint f=0; f<fTypes.size(); f++){
         string ft = fTypes[f];
         vector<int>allocableProducts =  MD->getAllocableProductIdsFromDeathTimber(regId, ft, dc, y, currentYear-1);
         iisskey key(y,regId,ft,dc);
         double deathTimber = MD->deathTimberInventory_get(key);
         double sum_total_st_allocable = 0;
         // Computing shares/weights or remaining st to allcate
         for(uint ap=0;ap<allocableProducts.size();ap++){
           sum_total_st_allocable += total_st.at(allocableProducts[ap]);
         }
         for(uint ap=0;ap<allocableProducts.size();ap++){
           double allocableShare = sum_total_st_allocable?total_st.at(allocableProducts[ap])/sum_total_st_allocable:0.0;
           double allocated = min(total_st[allocableProducts[ap]],deathTimber*allocableShare);
           MD->deathTimberInventory_incrOrAdd(key,-allocated);
           total_st[allocableProducts[ap]] -= allocated;
         }
       }
     }
   }
   return total_st;
}
```

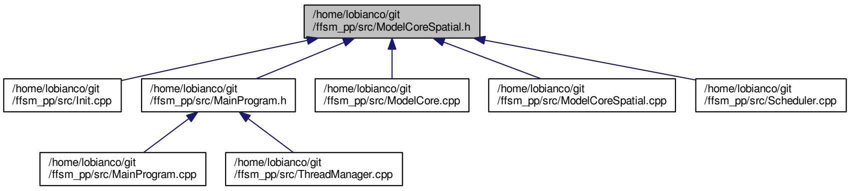
5.95 /home/lobianco/git/ffsm_pp/src/ModelCoreSpatial.h File Reference

```
#include <string>
#include <vector>
#include <map>
#include <stdexcept>
#include <iostream>
#include <sstream>
#include "BaseClass.h"
#include "ThreadManager.h"
#include "ModelData.h"
#include "Pixel.h"
```

Include dependency graph for ModelCoreSpatial.h:



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


/home/lobianco/git
/ffsm_pp/src/ModelCoreSpatial.h
/home/lobianco/git
/ffsm_pp/src/Init.cpp
/home/lobianco/git
/ffsm_pp/src/MainProgram.h
/home/lobianco/git
/ffsm_pp/src/ModelCore.cpp
/home/lobianco/git
/ffsm_pp/src/ModelCoreSpatial.cpp
/home/lobianco/git
/ffsm_pp/src/Scheduler.cpp
/home/lobianco/git
/ffsm_pp/src/MainProgram.cpp
/home/lobianco/git
/ffsm_pp/src/ThreadManager.cpp

Classes

- class ModelCoreSpatial

  The core of the model (spatial version).

5.96 ModelCoreSpatial.h

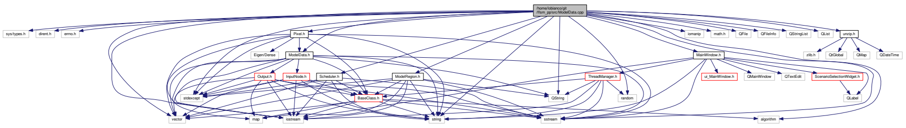
```
/******************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière                  *
 *   http://ffsm-project.org                                                  *
 *                                                                            *
 *   This program is free software; you can redistribute it and/or modify    *
 *   it under the terms of the GNU General Public License as published by    *
 *   the Free Software Foundation; either version 3 of the License, or       *
 *   (at your option) any later version, given the compliance with the       *
 *   exceptions listed in the file COPYING that is distributed together      *
 *   with this file.                                                          *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the        *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#ifndef MODELCORESPATIAL_H
#define MODELCORESPATIAL_H

// Core C++ headers
#include <string>
#include <vector>
#include <map>
#include <stdexcept>
#include <iostream>
#include <sstream>

// Qt headers...

// FFSM headers...
#include "BaseClass.h"
#include "ThreadManager.h"
#include "ModelData.h"
#include "Pixel.h"

/**
* \brief The core of the model (spatial version).
*
* Once the environment is initialised (mainly data load, space created), the model is run through the two functions runInitPeriod()
* and runSimulationYear().
*
*
*  Some importan notes:
*    V (volumes)    -> at the end of the year
*    In (inventary) -> at the beginning of the year
*    Area           -> at the end of the year
*    Harvesting     -> at the beginning of the year
*    Volumes are in Mm^3, Areas in the model in Ha (10000 m^2), in the layers in m^2, vHa in m^3/ha. Prices are in €/m^3.
*
*    BALANCE:
*    PROD_forLocal (sl) + PROD_forExp (sa) + IMP (da) + sum_reg(reg_trade_in) = CONS_fromLocal (dl) + CONS_fromImp (da) + EXP (sa) + sum_reg(reg_trade_out)
*    note that this means that sl includes alread reg_trade_out, and dl includes already reg_trade_in
*
* Where are volumes information ?
* - ip px->vol                                                         - by px, ft and dc
* - in forDataMap (through gft())                                      - by reg, ft and dc
* Where is area information ?
* - in px->area                                                        - by px, ft and dc
* - in forDataMap (through gft())                                      - by reg, ft and dc
* - in  px->values map (forArea_* layer, through px->getDoubleValue()) - by px and ft
*
* Aggregation of the Expected returns
*
* The problem is how to aggregate the expected returns, given at pixel anf ft level, first at the regional level, then at the ft group level (B/C) and
* total forest level and finally at national level from regional one.
*
* A - From pixel to region
* - weighted by total forest area in the pixel
* B1 - From ft to ft group
* - in each pixel we take the highest expRet within the pixel and we weight by farea to get the regional value
* B2 - From ft group to forest
* - actually, from ft to group: like b1, but we take the highest value in each px for any ft and we weight by forest area in the px to get the regional value
* C - From region to country
* - we weight the individual ft, ft group and forest by the different regional total forest areas.*
*
*/
class ModelCoreSpatial : public BaseClass {

public:
                        ModelCoreSpatial(ThreadManager* MTHREAD_h);
                        ~ModelCoreSpatial();

  void                  runInitPeriod();
  void                  runSimulationYear();

  void                  initMarketModule();       ///< computes st and pw for second year
```

```
and several needed-only-at-t0-vars for the market module
  void                  runMarketModule();        ///< computes st (supply total) and pw (weighted price). Optimisation inside.
  void                  runBiologicalModule();    ///< computes hV, hArea and new vol at end of year
  void                  runManagementModule();    ///< computes regArea and expectedReturns
  void                  sumRegionalForData();     ///< computes vol, hV, harvestedArea, regArea and expReturns at reg level from the pixel level
  void                  initialiseCarbonModule(); ///< call initialiseDeathBiomassStocks(), initialiseProductsStocks() and initialiseEmissionCounters()
  void                  initialiseDeathTimber();  ///< Set deathTimberInventory to zero for the previous years (under the hipotesis that we don't have advanced stock of death biomass usable as timber at the beginning of the simulation)

  void                  registerCarbonEvents();   ///< call registerHarvesting(), registerDeathBiomass(), registerProducts() and registerTransports()
  void                  cacheSettings();          ///< just cache exogenous settings from ModelData
  void                  initializePixelVolumes();  ///< distribuite regional exogenous volumes to pixel volumes using corine land cover area as weight
  void                  assignSpMultiplierPropToVols(); // assign the spatial multiplier (used in the time of return) based no more on a Normal distribution but on the volumes present in the pixel: more volume, more the pixel is fit for the ft
  void                  initializePixelArea();    ///< compute px->area for each ft and dc
  void                  resetPixelValues();       ///< swap volumes->lagged_volumes and reset the other pixel vectors
  void                  cachePixelExogenousData();///< computes pixel level tp, meta and mort
  void                  computeInventary();       ///< in=f(vol_t-1)
  void                  computeCumulativeData();  ///< computes cumTp_exp, vHa_exp, vHa
  void                  updateMapAreas();         ///< computes forArea_{ft}
  void                  updateOtherMapData();     ///< update (if the layer exists) other gis-based data, as volumes and expected returns, taking them from the data in the px object
  double                computeExpectedPrice(const double & curLocPrice, const double & worldCurPrice, const double & worldFutPrice, const double & sl, const double & sa, const double & expCoef); ///< Compute weighted expected price for a given product.
  void                  printDebugInitRegionalValues(); ///< print initial inv, st, sl and sa in each region
  vector <double>       allocateHarvesting(vector<double> total_st, const int & regId); ///< Using the deathTimberInventory map, this function allocate the total st in st from death timber (that goes reduce the deathTimberInventory map) and stFromHarvesting that is what it remains after the allocation to death timber.
  void                  loadExogenousForestLayers(const string & what); ///< Set pixel volumes (what="vol") OR areas (what="area") by specific forest types as defined in gis layers for volumes and proportionally to volumes for areas.

  // convenient handles to equivalent ModelData functions..
  double              gpd(const string &type_h, const int& regId_h, const string &prodId_h, const int& year=DATA_NOW, const string &freeDim_h="") const {return MTHREAD->MD->getProdData(type_h, regId_h, prodId_h, year, freeDim_h);};
  double              gfd(const string &type_h, const int& regId_h, const string &forType_h, const string &freeDim_h, const int& year=DATA_NOW) const {return MTHREAD->MD->getForData(type_h, regId_h, forType_h, freeDim_h, year);};
  void                spd(const double& value_h, const string &type_h, const int& regId_h, const string &prodId_h, const int& year=DATA_NOW, const bool& allowCreate=false, const string &freeDim_h="") const {MTHREAD->MD->setProdData(value_h, type_h, regId_h, prodId_h, year, allowCreate, freeDim_h);};
  void                sfd(const double& value_h, const string &type_h, const int& regId_h, const string &forType_h, const string &freeDim_h, const int& year=DATA_NOW, const bool& allowCreate=false) const {MTHREAD->MD->setForData(value_h, type_h, regId_h, forType_h, freeDim_h, year, allowCreate);};
  bool                app(const string &prod_h, const string &forType_h, const string &dClass_h) const {return MTHREAD->MD->assessProdPossibility(prod_h, forType_h, dClass_h);};

private:
  ModelData* MD;
  int firstYear;
  int secondYear;
  int thirdYear;
  int WL2;
  vector <int> regIds2;
  vector <string> priProducts;
  vector <string> secProducts;
  vector <string> allProducts;
  vector <string> dClasses;
  vector <string> pDClasses;
  vector <string> fTypes;
  vector <vector <int> > l2r;
  string regType;
  string natRegAllocation;
  //double mr;
  vector <Pixel*> regPx; // pixels behaving to the current region
  bool rescaleFrequencies;
  bool oldVol2AreaMethod;
  string forestAreaChangeMethod;
  double ir; // interest rate
};

#endif // MODELCORESPATIAL_H
```

5.97 /home/lobianco/git/ffsm_pp/src/ModelData.cpp File Reference

```
#include <sys/types.h>
#include <dirent.h>
#include <errno.h>
#include <iostream>
#include <vector>
#include <string>
#include <sstream>
#include <stdexcept>
#include <algorithm>
#include <iomanip>
#include <math.h>
#include <random>
#include <QFile>
#include <QFileInfo>
#include <QString>
#include <QStringList>
#include <QList>
#include "unzip.h"
#include "ModelData.h"
#include "MainWindow.h"
#include "Scheduler.h"
#include "ModelRegion.h"
#include "Pixel.h"
```

Include dependency graph for ModelData.cpp:



Typedefs

- typedef map< string, vector< double > > DataMap
- typedef pair< string, vector< double > > DataPair

5.97.1 Typedef Documentation

5.97.1.1 typedef map<string, vector <double> > DataMap

Definition at line 56 of file ModelData.cpp.

5.97.1.2 typedef pair<string, vector <double> > DataPair

Definition at line 57 of file ModelData.cpp.

5.98 ModelData.cpp

```
/****************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière                *
 *   http://ffsm-project.org                                                *
 *                                                                          *
 *   This program is free software; you can redistribute it and/or modify   *
 *   it under the terms of the GNU General Public License as published by   *
 *   the Free Software Foundation; either version 3 of the License, or      *
 *   (at your option) any later version, given the compliance with the      *
 *   exceptions listed in the file COPYING that is distribued together      *
 *   with this file.                                                        *
 *                                                                          *
 *   This program is distributed in the hope that it will be useful,        *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of         *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the          *
 *   GNU General Public License for more details.                           *
 *                                                                          *
 *   You should have received a copy of the GNU General Public License      *
 *   along with this program; if not, write to the                          *
 *   Free Software Foundation, Inc.,                                        *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.              *
 ****************************************************************************/

#include <sys/types.h>
#include <dirent.h>
#include <errno.h>
#include <iostream>

#include <vector>
#include <string>
#include <sstream>
#include <stdexcept>
#include <algorithm> //alghoritm used to reverse an array ( reverse(v.begin(), v.end()); )
#include <iomanip> // for unzip
#include <math.h>
#include <random> // for random temp directory to unzip

// Qt headers..
#include <QFile>
#include <QFileInfo>
#include <QString>
#include <QStringList>
#include <QList>

// Unzip headers..
#include "unzip.h"

// RegMAS headers..
#include "ModelData.h"
//#include "InputDocument.h"
//#include "InputNode.h"
#include "MainWindow.h"
#include "Scheduler.h"
#include "ModelRegion.h"
#include "Pixel.h"

typedef map<string, vector <double> > DataMap;
typedef pair<string, vector <double> > DataPair;



ModelData::ModelData(ThreadManager* MTHREAD_h){
  MTHREAD = MTHREAD_h;
  errorLevel = MSG_ERROR;
}

ModelData::~ModelData(){

}

forType*
ModelData::getForType(string &forTypeId_h){
  for(int i=0;i<forTypes.size();i++){
    if(forTypes[i].forTypeId==forTypeId_h) return &forTypes[i];
  }
  msgOut(MSG_CRITICAL_ERROR,"forTypeId "+forTypeId_h+" not found. Aborting.");
}

int
ModelData::getForTypeCounter(string& forTypeId_h, bool all){
  vector <string> fTIds = getForTypeIds(all);
  for(int i=0;i<fTIds.size();i++){
    if(fTIds[i]==forTypeId_h) return i;
  }
  msgOut(MSG_CRITICAL_ERROR,"forTypeId "+forTypeId_h+" not found in "+((string)
__func__ )+". Aborting.");
}

string
ModelData::getForTypeParentId(const string &forTypeId_h){
    for(int i=0;i<forTypes.size();i++){
        if(forTypes[i].forTypeId==forTypeId_h) return forTypes[i].ereditatedFrom;
    }
    msgOut(MSG_CRITICAL_ERROR,"forTypeId "+forTypeId_h+" not found. Aborting.");
}

vector<string>
ModelData::getForTypeChilds(const string &forTypeId_h){
  vector<string> childs;
  for(int i=0;i<forTypes.size();i++){
      if(forTypes[i].ereditatedFrom==forTypeId_h) {
          childs.push_back(forTypes[i].forTypeId);
      }
  }
  return childs;
}

vector<int>
ModelData::getForTypeChilds_pos(const string &forTypeId_h, bool all){
  vector <int> childs;
  vector <string> fTIds = getForTypeIds(all);
  for(int i=0;i<fTIds.size();i++){
      forType* ft = getForType(fTIds[i]);
      if(ft->ereditatedFrom==forTypeId_h) {
          childs.push_back(i);
      }
  }
  return childs;
}

vector<string>
ModelData::getForTypeParents(){
  vector<string> parents;
  for(int i=0;i<forTypes.size();i++){
    string parent = forTypes[i].ereditatedFrom;
    if(!inVector(parent,parents) && parent != ""){
      parents.push_back(parent);
    }
  }
  return parents;
}


int
ModelData::getNForTypesChilds(const string& forTypeId_h){
    int nChilds = 0;
    for(int i=0;i<forTypes.size();i++){
        if(forTypes[i].ereditatedFrom==forTypeId_h) {
            nChilds ++;
        }
    }
    return nChilds;
}

vector<string>
ModelData::getScenarios(){
  vector<string> toReturn;
  LLData table = getTable("scenarios");
  for(int i=0;i<table.nrecords();i++){
    string scenarioName = table.getData(i,"id");
    toReturn.push_back(scenarioName);
  }
  return toReturn;
}

int
ModelData::getScenarioIndex(){
  vector<string> scenarios = getScenarios(); /// \todo Check that I can call this
 function all around the model and not only at the beginning
  string currentScenario = MTHREAD->getScenarioName();
  for(int i=0;i<scenarios.size();i++){
    if (currentScenario == scenarios[i]){
      return i;
    }
  }
  msgOut(MSG_CRITICAL_ERROR, "function getScenarioIndex() didn't found the current
 scenarioName within those returned by getScenarios().");
  return 0;
}

void
ModelData::setScenarioData(){
  LLData table = getTable("scenarios");
  for(int i=0;i<table.nrecords();i++){
    string recordScenarioName = table.getData(i,"id");
    if (recordScenarioName == MTHREAD->getScenarioName()){
      scenario.id             = recordScenarioName;
      scenario.shortDesc      = table.getData(i,"shortDesc");
      scenario.longDesc       = table.getData(i,"longDesc");
      scenario.settingTable   = table.getData(i,"settingTable");
      scenario.forDataTable   = table.getData(i,"forDataTable");
      scenario.prodDataTable  = table.getData(i,"prodDataTable");
      scenario.forToProdTable = table.getData(i,"forToProdTable");
      scenario.pathTable      = table.getData(i,"pathTable");
      return;
    }
  }


}

void
ModelData::setDefaultSettings(){

  LLData table = getTable("settings");
  int nheaders = table.nheaders();
  for (int i=0; i< table.nrecords();i++){
    BasicData SETT;
    SETT.name = table.getData(i,"name");
    string type = table.getData(i,"type");
    SETT.type = getType(type);
    SETT.comment = table.getData(i,"comment");
    vector <string> values;
    for (int z=0;z<nheaders-3;z++){ // don't consider name, type and comment headers
      string toSearch = "value_"+i2s(z);
      string value = table.getData(i,toSearch);
      if (value != ""){
        values.push_back(value);
      }
    }
    SETT.values = values;
    programSettingsVector.push_back(SETT);
  }

  msgOut(MSG_INFO,"### USING SCENARIO: "+MTHREAD->
getScenarioName()+" ###");

  setOutputDirectory(getStringSetting("outputDirname").c_str());
}

void
ModelData::setScenarioSettings(){

  if(scenario.settingTable=="") {return;}
  LLData table = getTable(scenario.settingTable,
MSG_CRITICAL_ERROR); //this scenario could not have an associated setting sheet

  int nheaders = table.nheaders();
  for(int i=0; i< table.nrecords(); i++){
    BasicData SETT;
    string name = table.getData(i,"name");
    string stype = table.getData(i,"type");
    int type = getType(stype);
    string comment = table.getData(i,"comment");
    vector <string> values;
    for (int z=0;z<nheaders-3;z++){ // don't consider name, type and comment headers
      string toSearch = "value_"+i2s(z);
      string value = table.getData(i,toSearch);
      if (value != ""){
        values.push_back(value);
      }
    }

    for(uint i=0;i<programSettingsVector.size();i++){
      if(programSettingsVector[i].name == name){
        programSettingsVector[i].values = values;
        programSettingsVector[i].type = type;
        programSettingsVector[i].comment = comment;
        break;
      }
    }

  }

  setOutputDirectory(getStringSetting("outputDirname").c_str());
}

void
ModelData::addSetting(string name_h, vector <string> values_h, int type_h, string
comment_h){

  for (uint i=0;i<programSettingsVector.size();i++){
    if (programSettingsVector.at(i).name == name_h){
      msgOut(MSG_ERROR, "I already have setting "+name_h+".. Nothing is added..");
      return;
    }
  }
  BasicData SETT;
  SETT.name = name_h;
  SETT.values = values_h;
  SETT.type= type_h;
  SETT.comment = comment_h;
  programSettingsVector.push_back(SETT);
}

void
ModelData::addSetting(string name_h, string value_h, int type_h, string comment_h){
  vector <string> values;
  values.push_back(value_h);
  addSetting(name_h, values, type_h, comment_h);
}

void
ModelData::cacheSettings(){
  cached_initialYear = getIntSetting("initialYear");
  diamClasses = getStringVectorSetting("dClasses");
  priProducts = getStringVectorSetting("priProducts");
  secProducts = getStringVectorSetting("secProducts");
  allProducts = priProducts;
  allProducts.insert( allProducts.end(), secProducts.begin(),
secProducts.end() );
}

// ################################################################################

void
ModelData::createRegions(){
  // first create regions and assign basic data...
  LLData table = getTable("regions");
  for (int i=0; i< table.nrecords();i++){
    ModelRegion REGION(MTHREAD,
                 s2i(table.getData(i,"regId")),
                 table.getData(i,"regSName"),
                 table.getData(i,"regLName"),
                 s2i(table.getData(i,"regLevel")),
                 s2i(table.getData(i,"parRegId")),
                 s2b(table.getData(i,"isResidual")));
    regionsVector.push_back(REGION);
  }
  // Now let's assign the parent/children pointers..
  for (int i=0; i< regionsVector.size();i++){
    // let's assign the parent:
    regionsVector[i].setParent(this->getRegion(
regionsVector[i].getParRegId()));
    // let's assign the children:
    vector<ModelRegion*> kids;
    for (int y=0; y< regionsVector.size();y++){
      if(regionsVector[y].getParRegId() == regionsVector[i].getRegId() ){
        kids.push_back(&regionsVector[y]);
      }
    }
    regionsVector[i].setChildren(kids);
  }
}

ModelRegion*
ModelData::getRegion(int regId_h){
  for (int i=0; i< regionsVector.size();i++){
    if(regionsVector[i].getRegId()==regId_h){
      return &regionsVector[i];
    }
  }
  msgOut(MSG_CRITICAL_ERROR, "Region id "+i2s(regId_h)+" not found, check your
 input data. Aborting simulation.");
}

bool
ModelData::regionExist (const int & regId_h) const {
  for (int i=0; i< regionsVector.size();i++){
    if(regionsVector[i].getRegId()==regId_h){
      return true;
    }
  }
  return false;
}

```

```
vector <int>
ModelData::getRegionIds(int level_h, bool excludeResidual){
  vector <int> toReturn;
  for(uint i=0;i<regionsVector.size();i++){
    if(regionsVector[i].getRegLevel()==level_h){
      if( (!excludeResidual) || (!regionsVector[i].getIsResidual())){
        toReturn.push_back(regionsVector[i].getRegId());
      }
    }
  }
  return toReturn;
}

vector <ModelRegion*>
ModelData::getAllRegions(bool excludeResidual){
  vector <ModelRegion*> toReturn;
  for(uint i=0;i<regionsVector.size();i++){
    if( (!excludeResidual) || (!regionsVector[i].getIsResidual())){
      toReturn.push_back(&regionsVector[i]);
    }
  }
  return toReturn;
}

vector < vector <int> >
ModelData::getRegionIds( bool excludeResidual){
  vector < vector <int> > toReturn;
  vector <int> l1regIds =  MTHREAD->MD->getRegionIds(1, excludeResidual);
  for(uint i=0;i<l1regIds.size();i++){
    vector<int> l2ChildrenIds;
    ModelRegion* l1Region = MTHREAD->MD->getRegion(l1regIds[i]);
    vector<ModelRegion*> l2Childrens = l1Region->getChildren(excludeResidual);
    for(uint j=0;j<l2Childrens.size();j++){
      l2ChildrenIds.push_back(l2Childrens[j]->getRegId());
    }
    if(l2ChildrenIds.size()){
      toReturn.push_back(l2ChildrenIds);
    }
  }
  return toReturn;
}

string
ModelData::regId2RegSName (const int & regId_h) const {
  ModelRegion* reg = MTHREAD->MD->getRegion(regId_h);
  return reg->getRegSName();
}

int
ModelData::regSName2RegId (const string & regSName_h) const{
  ModelRegion* reg;
  for(uint i=0; i<3; i++){
    vector <int> regIds =  MTHREAD->MD->getRegionIds(i, false);
    for(uint j=0;j<regIds.size();j++){
      reg = MTHREAD->MD->getRegion(regIds[j]);
      if(reg->getRegSName()==regSName_h) {return regIds[j];}
    }
  }
  msgOut(MSG_CRITICAL_ERROR,"Regional short name not found.");
}




vector <string>
ModelData::getForTypeIds(bool all){
  vector <string> toReturn;
  for(uint i=0;i<forTypes.size();i++){
    if(forTypes[i].memType!=1 || all) {
      toReturn.push_back(forTypes[i].forTypeId);
    }
  }
  return toReturn;
}

const bool
ModelData::assessProdPossibility(const string &prod_h, const string &
forType_h, const string &dClass_h){
  bool ok=false;
  for(uint i=0;i<forToProdVector.size();i++){
    if(    forToProdVector[i].product == prod_h
      && forToProdVector[i].forType == forType_h
      && forToProdVector[i].dClass  == dClass_h
    ){
      return true;
    }
  }
  return false;
}


const int
ModelData::getMaxYearUsableDeathTimber(){
  int maxMaxYears = 0;
  for(uint i=0;i<forToProdVector.size();i++){
    if(forToProdVector[i].maxYears > maxMaxYears){
      maxMaxYears = forToProdVector[i].maxYears;
    }
  }
  return maxMaxYears;
}


const int
ModelData::getMaxYearUsableDeathTimber(const string &prod_h, const string &forType_h, const string &dClass_h){
  for(uint i=0;i<forToProdVector.size();i++){
    if(    forToProdVector[i].product == prod_h
      && forToProdVector[i].forType == forType_h
      && forToProdVector[i].dClass  == dClass_h
    ){
      return forToProdVector[i].maxYears;
    }
  }
  msgOut(MSG_CRITICAL_ERROR,"In getMaxYearUsableDeathTimber() I has been asked of a combination that I don't know how to handle.");
}

void
ModelData::setDefaultForData(){
  msgOut(MSG_DEBUG,"Loading forest sector data..");
  LLData table = getTable("forData");
  int nheaders = table.nheaders();
  for (int i=0; i< table.nrecords();i++){
    vector <double> values;
    for (int z=0;z<nheaders-4;z++){ // don't consider parName, region, forType and diamClass headers
      string toSearch = "value_"+i2s(z);
      string value = table.getData(i,toSearch);
      if (value != ""){
        values.push_back(s2d(value));
      }
    }
        string keys = makeKeyForData(table.getData(i,"parName"), table.getData(i,"region"),table.getData(i,"forType"),table.getData(i,"freeDim"));
    forDataMap.insert(std::pair<string, vector<double> >(keys, values));
  }
}

void
ModelData::setScenarioForData(){

  if(scenario.forDataTable==""){return;}
  LLData table = getTable(scenario.forDataTable, MSG_CRITICAL_ERROR);

  int nheaders = table.nheaders();
  for(int i=0; i< table.nrecords(); i++){
    bool found = false;
        string key = makeKeyForData(table.getData(i,"parName"),table.getData(i,"region"),table.getData(i,"forType"),table.getData(i,"freeDim"));
    vector <double> values;
    for (int z=0;z<nheaders-4;z++){ // don't consider parName, region, forType and diamClass headers
      string toSearch = "value_"+i2s(z);
      string value = table.getData(i,toSearch);
      if (value != ""){
        values.push_back(s2d(value));
      }
    }
    map <string, vector < double > >::iterator p;
    p=forDataMap.find(key);
    if(p != forDataMap.end()) {
      // updating an existing record
      p->second = values;
    }
    else {
      // new one, adding it
      forDataMap.insert(std::pair<string, vector<double> >(key, values));
    }
  }
}

void
ModelData::setDefaultProdData(){

  msgOut(MSG_DEBUG,"Loading products data..");
  LLData table = getTable("prodData");
  int nheaders = table.nheaders();

  for (int i=0; i< table.nrecords();i++){
//    prodData PDATA;
//    PDATA.parName = table.getData(i,"parName");
//    PDATA.region = s2i(table.getData(i,"region"));
//    PDATA.prod = table.getData(i,"prod");
//    PDATA.freeDim = table.getData(i,"freeDim");
    vector <double> values;
    for (int z=0;z<nheaders-4;z++){ // don’t consider parName, region, prod and freeDim headers
      string toSearch = "value_"+i2s(z);
      string value = table.getData(i,toSearch);
      if (value != ""){
        values.push_back(s2d(value));
      }
    }
//    PDATA.values = values;
//    prodDataVector.push_back(PDATA);
    string keys = makeKeyProdData(table.getData(i,"parName"), table.
getData(i,"region"),table.getData(i,"prod"),table.getData(i,"freeDim"));
    prodDataMap.insert(std::pair<string, vector<double> >(keys, values));
    //giving a link to it to its own region:
//    getRegion(PDATA.region)->addProdData(&PDATA);
  }
}

void
ModelData::setScenarioProdData(){

  if(scenario.prodDataTable==""){return;}
  LLData table = getTable(scenario.prodDataTable,
MSG_CRITICAL_ERROR); //this scenario could not have an associated setting sheet

  int nheaders = table.nheaders();
  for(int i=0; i< table.nrecords(); i++){
    //prodData PDATA;
    bool found = false;
    string key = makeKeyProdData(table.getData(i,"parName"),table.
getData(i,"region"),table.getData(i,"prod"),table.getData(i,"freeDim"));

    //PDATA.parName = table.getData(i,"parName");
    //PDATA.region = s2i(table.getData(i,"region"));
    //PDATA.prod = table.getData(i,"prod");
    //PDATA.freeDim = table.getData(i,"freeDim");
    vector <double> values;
    for (int z=0;z<nheaders-4;z++){// don’t consider parName, region, prod and freeDim headers
      string toSearch = "value_"+i2s(z);
      string value = table.getData(i,toSearch);
      if (value != ""){
        values.push_back(s2d(value));
      }
    }
    //PDATA.values = values;
    //for(uint i=0;i<prodDataVector.size();i++){
    //  if(prodDataVector[i].parName == PDATA.parName
    //     && prodDataVector[i].region == PDATA.region
    //     && prodDataVector[i].prod == PDATA.prod
    //     && prodDataVector[i].freeDim == PDATA.freeDim){
    //    // existing prodData..
    //    prodDataVector[i].values = PDATA.values;
    //    found = true;
    //    break;
    //  }
    //}
    //if(!found){
    //  // new one, adding it
    //  prodDataVector.push_back(PDATA);
    //  //giving a link to it to its own region:
    //  getRegion(PDATA.region)->addProdData(&PDATA);
    //}

    map <string, vector < double > >::iterator p;
    p=prodDataMap.find(key);
    if(p != prodDataMap.end()) {
      // updating an existing record
      p->second = values;
    }
    else {
      // new one, adding it
      prodDataMap.insert(std::pair<string, vector<double> >(key, values));
    }
  }
}

void
```

```
ModelData::setDefaultProductResourceMatrixLink(){
  msgOut(MSG_DEBUG,"Loading forest resource to primary products io matrix..");
  LLData table = getTable("forToProd");
  for (int i=0; i< table.nrecords();i++){
    forToProd F2PDATA;
    F2PDATA.product = table.getData(i,"product");
    F2PDATA.forType = table.getData(i,"forType");
    F2PDATA.dClass = table.getData(i,"dClass");
    F2PDATA.maxYears = s2i(table.getData(i,"maxYears"));
    forToProdVector.push_back(F2PDATA);
  }
}

void
ModelData::setScenarioProductResourceMatrixLink(){
  if(scenario.forToProdTable==""){return;}
  LLData table = getTable(scenario.forToProdTable,
MSG_CRITICAL_ERROR); //this scenario could not have an associated setting sheet

  int nheaders = table.nheaders();
  forToProdVector.clear();
  for (int i=0; i< table.nrecords();i++){
    forToProd F2PDATA;
    F2PDATA.product = table.getData(i,"product");
    F2PDATA.forType = table.getData(i,"forType");
    F2PDATA.dClass = table.getData(i,"dClass");
    forToProdVector.push_back(F2PDATA);
  }
}

void
ModelData::setForestTypes(){
  LLData table = getTable("forTypes");
  for (int i=0; i< table.nrecords();i++){
    forType FTYPE;
    FTYPE.forTypeId = table.getData(i,"forTypeId");
    FTYPE.forLabel = table.getData(i,"forLabel");
    FTYPE.memType = s2i(table.getData(i,"memType"));
    FTYPE.forLayer = table.getData(i,"forLayer");
    FTYPE.ereditatedFrom = table.getData(i,"ereditatedFrom");
    if(FTYPE.memType == 3 && !getBoolSetting("useSpExplicitForestTypes")) continue;
    forTypes.push_back(FTYPE);
  }
}

void
ModelData::setReclassificationRules(){

  msgOut(MSG_DEBUG,"Loading (but not yet applying) reclassification rules..");
  LLData table = getTable("reclRules");
  for (int i=0; i< table.nrecords();i++){
    reclRule RL;
    RL.regId = s2i(table.getData(i,"regID"));
    RL.forTypeIn = table.getData(i,"forTypeIn");
    RL.forTypeOut = table.getData(i,"forTypeOut");
    RL.coeff = s2d(table.getData(i,"coeff"));
    reclRules.push_back(RL);
  }
}

void
ModelData::setDefaultPathogenRules(){

  if(!getBoolSetting("usePathogenModule")) return;
  msgOut(MSG_DEBUG,"Loading pathogen rules..");
  LLData table = getTable("pathRules");
  int nheaders = table.nheaders();
  for (int i=0; i< table.nrecords();i++){
    pathRule PR;
    PR.forType = table.getData(i,"forType");
    PR.dClass = table.getData(i,"dClass");
    PR.pathId = table.getData(i,"path_name");
    PR.pres_min = s2d(table.getData(i,"pres_min"));

    vector <double> values;
    for (int z=0;z<nheaders-4;z++){ // don't consider forType, dClass, path_name and pres_min  headers
      string toSearch = "year_"+i2s(z);
      string value = table.getData(i,toSearch);
      if (value != ""){
        values.push_back(s2d(value));
      }
    }
    PR.mortCoefficents = values;

    pathRules.push_back(PR);
  }
}
```

```

void
ModelData::setScenarioPathogenRules(){

  if(scenario.pathTable==""){return;}
  LLData table = getTable(scenario.pathTable,
MSG_CRITICAL_ERROR); //this scenario could not have an associated setting sheet

  int nheaders = table.nheaders();
  for (int i=0; i< table.nrecords();i++){
    pathRule PR;
    PR.forType = table.getData(i,"forType");
    PR.dClass = table.getData(i,"dClass");
    PR.pathId = table.getData(i,"path_name");
    PR.pres_min = s2d(table.getData(i,"pres_min"));

    vector <double> values;
    for (int z=0;z<nheaders-4;z++){ // don't consider forType, dClass, path_name and pres_min  headers
      string toSearch = "year_"+i2s(z);
      string value = table.getData(i,toSearch);
      if (value != ""){
        values.push_back(s2d(value));
      }
    }
    PR.mortCoefficents = values;

    bool found = false;
    for(uint i=0;i<pathRules.size();i++){
      if(    pathRules[i].forType == PR.forType
          && pathRules[i].dClass == PR.dClass
          && pathRules[i].pathId == PR.pathId
         ){
        pathRules[i].pres_min = PR.pres_min;
        pathRules[i].mortCoefficents = PR.mortCoefficents;
        found = true;
        break;
      }
    }
    if(!found){
      pathRules.push_back(PR);
    }
  } // end for each table record
}

/// Cancel all reg1 level data and trasform them in reg2 level if not already existing
void
ModelData::applyOverrides(){

  if(!getBoolSetting("applyOverriding")) return;
  msgOut(MSG_INFO, "Starting regional overriding analysis..");

  DataMap::iterator p;
  string parName,prod,freeDim,forType,diamClass, key;
  int regId;
  DataMap toBeAdded;
  vector <string> keysToRemove;


  //apply override from level 0 to level 1 for forestry data
  toBeAdded.clear();
  keysToRemove.clear();
  for(p=forDataMap.begin();p!=forDataMap.end();p++){
    unpackKeyForData(p->first,parName,regId,forType,diamClass);
    //if(!regionExist(regId)) continue;
    if(getRegion(regId)->getRegLevel() == 0){
      vector<ModelRegion*> childs = getRegion(regId)->getChildren(false);
      for(uint j=0;j<childs.size();j++){
        bool found = false;
        key = makeKeyForData(parName,i2s(childs[j]->getRegId()),forType,diamClass);
        if (!dataMapCheckExist(forDataMap,key,true)){
          toBeAdded.insert(DataPair(key,p->second));
        }
      }
      keysToRemove.push_back(p->first);
    }
  }
  forDataMap.insert(toBeAdded.begin(),toBeAdded.end());
  for(uint i=0;i<keysToRemove.size();i++){
    DataMap::iterator rem = forDataMap.find(keysToRemove[i]);
    if(rem != forDataMap.end()){
      forDataMap.erase(rem);
    }
  }




  //apply override from level 1 to level 2 for forestry data
  toBeAdded.clear();
  keysToRemove.clear();
  for(p=forDataMap.begin();p!=forDataMap.end();p++){
    unpackKeyForData(p->first,parName,regId,forType,diamClass);
    //if(!regionExist(regId)) continue;
    if(getRegion(regId)->getRegLevel() == 1){
      vector<ModelRegion*> childs = getRegion(regId)->getChildren(false);
      for(uint j=0;j<childs.size();j++){
        bool found = false;
        key = makeKeyForData(parName,i2s(childs[j]->getRegId()),forType,diamClass);
        if (!dataMapCheckExist(forDataMap,key,true)){
          toBeAdded.insert(DataPair(key,p->second));
        }
      }
      keysToRemove.push_back(p->first);
    }
  }
  forDataMap.insert(toBeAdded.begin(),toBeAdded.end());
  for(uint i=0;i<keysToRemove.size();i++){
    DataMap::iterator rem = forDataMap.find(keysToRemove[i]);
    if(rem != forDataMap.end()){
      forDataMap.erase(rem);
    }
  }

  //apply override from level 0 to level 1 for production data
  toBeAdded.clear();
  keysToRemove.clear();
  for(p=prodDataMap.begin();p!=prodDataMap.end();p++){
    unpackKeyProdData(p->first,parName,regId,prod,freeDim);
    //if(!regionExist(regId)) continue;
    if(getRegion(regId)->getRegLevel() == 0){
      vector<ModelRegion*> childs = getRegion(regId)->getChildren(false);
      for(uint j=0;j<childs.size();j++){
        bool found = false;
        key = makeKeyProdData(parName,i2s(childs[j]->getRegId()),prod,freeDim);
        if (!dataMapCheckExist(prodDataMap,key,true)){
          toBeAdded.insert(DataPair(key,p->second));
        }
      }
      //prodDataMap.erase(p);
      //p--;
      keysToRemove.push_back(p->first);
    }
  }
  prodDataMap.insert(toBeAdded.begin(),toBeAdded.end());
  for(uint i=0;i<keysToRemove.size();i++){
    DataMap::iterator rem = prodDataMap.find(keysToRemove[i]);
    if(rem != prodDataMap.end()){
      prodDataMap.erase(rem);
    }
  }


  //apply override from level 1 to level 2 for production data
  toBeAdded.clear();
  keysToRemove.clear();
  for(p=prodDataMap.begin();p!=prodDataMap.end();p++){
    string debug = p->first;
    unpackKeyProdData(p->first,parName,regId,prod,freeDim);
    //if(!regionExist(regId)) continue;
    if(getRegion(regId)->getRegLevel() == 1){
      vector<ModelRegion*> childs = getRegion(regId)->getChildren(false);
      for(uint j=0;j<childs.size();j++){
        bool found = false;
        key = makeKeyProdData(parName,i2s(childs[j]->getRegId()),prod,freeDim);
        if (!dataMapCheckExist(prodDataMap,key,true)){
          toBeAdded.insert(DataPair(key,p->second));
        }
      }
      //prodDataMap.erase(p);
      //p--;
      keysToRemove.push_back(p->first);
    }
  }
  prodDataMap.insert(toBeAdded.begin(),toBeAdded.end());
    for(uint i=0;i<keysToRemove.size();i++){
    DataMap::iterator rem = prodDataMap.find(keysToRemove[i]);
    if(rem != prodDataMap.end()){
      prodDataMap.erase(rem);
    }
  }

  //apply override from level 0 to level 1 for reclassification rules
  for(uint i=0;i<reclRules.size();i++){
    if(reclRules[i].regId == 0){
      //if(!regionExist(reclRules[i].regId)) continue;
      for(uint j=0;j<getRegion(reclRules[i].regId)->getNChildren(false);j++){
        vector<ModelRegion*> childs = getRegion(reclRules[i].regId)->getChildren(false);
        bool found = 0;
        for(uint z=0;z<reclRules.size();z++){
          if(   reclRules[z].regId == childs[j]->getRegId()
             && reclRules[z].forTypeIn == reclRules[i].forTypeIn
             && reclRules[z].forTypeOut == reclRules[i].forTypeOut
          ){
            found = true; // do nothing, this child has been already manually overritten
            break;
          }
        }
        if(!found){
          reclRule RR;
          RR.regId      = childs[j]->getRegId();
          RR.forTypeIn  = reclRules[i].forTypeIn;
          RR.forTypeOut = reclRules[i].forTypeOut;
          RR.coeff      = reclRules[i].coeff;
          reclRules.push_back(RR);
        }
      }
      reclRules.erase(reclRules.begin()+i);
      i--;
    }
  }

  //apply override from level 1 to level 2 for reclassification rules
  for(uint i=0;i<reclRules.size();i++){
    //if(!regionExist(reclRules[i].regId)) continue;
    if(getRegion(reclRules[i].regId)->getRegLevel() == 1){
      for(uint j=0;j<getRegion(reclRules[i].regId)->getNChildren(false);j++){
        vector<ModelRegion*> childs = getRegion(reclRules[i].regId)->getChildren(false);
        bool found = 0;
        for(uint z=0;z<reclRules.size();z++){
          if(   reclRules[z].regId == childs[j]->getRegId()
             && reclRules[z].forTypeIn == reclRules[i].forTypeIn
             && reclRules[z].forTypeOut == reclRules[i].forTypeOut
          ){
            found = true; // do nothing, this child has been already manually overritten
            break;
          }
        }
        if(!found){
          reclRule RR;
          RR.regId      = childs[j]->getRegId();
          RR.forTypeIn  = reclRules[i].forTypeIn;
          RR.forTypeOut = reclRules[i].forTypeOut;
          RR.coeff      = reclRules[i].coeff;
          reclRules.push_back(RR);
        }
      }
      reclRules.erase(reclRules.begin()+i);
      i--;
    }
  }
}

/**
The applyDebugMode flag all level2 regions not in the "debugRegions" option as "residual" (so they are in the map but not in the model code) and remove the primary and secondary products that are not included in the debugPriProducts and debugSecProducts options.
*/
void
ModelData::applyDebugMode(){
  if(! getBoolSetting("debugFlag")) return;

  vector <int> debugRegions = getIntVectorSetting("debugRegions");
  vector <string> debugPriProducts = getStringVectorSetting("debugPriProducts");
  vector <string> debugSecProducts = getStringVectorSetting("debugSecProducts");

  for(uint i=0;i< regionsVector.size();i++){
    if (regionsVector[i].getRegLevel()==2){
      bool found= false;
      for(uint j=0;j<debugRegions.size();j++){
        if (debugRegions[j] == regionsVector[i].getRegId()){
          found = true;
          break;
        }
      }
      if(!found){ // not in the list to keep
        regionsVector[i].setIsResidual(true);
      }
    }
  }

  for (uint i=0; i<programSettingsVector.size();i++){
    if (programSettingsVector.at(i).name == "priProducts"){
      programSettingsVector.at(i).values = debugPriProducts;
    } else if (programSettingsVector.at(i).name == "secProducts"){
      programSettingsVector.at(i).values = debugSecProducts;
    }
  }

}

void
ModelData::setOutputDirectory(const char* output_dirname_h){

  if (strlen(output_dirname_h)==0){
    outputDirname=baseDirectory+"output/";
  }
  else {
    outputDirname=output_dirname_h;
  }
  MTHREAD->setOutputDirName(outputDirname); //for the GUI
}

string
ModelData::getBaseData (const string &name_h, int type_h, int position){
  // If the data is called with DATA_NOW we interpret the array of values as a temporal array and we return the value at the current time.
  if(position == DATA_NOW) {
    position = MTHREAD->SCD->getIteration();
  }
  for (uint i=0; i<programSettingsVector.size();i++){
    if (programSettingsVector.at(i).name == name_h){
      int type = programSettingsVector.at(i).type;
      if(type != type_h){msgOut(MSG_CRITICAL_ERROR, "mismatching type in calling getBaseData() for "+name_h);}
      if(programSettingsVector.at(i).values.size() > ((uint)position)) {
        return programSettingsVector.at(i).values.at(position);
      } else if (programSettingsVector.at(i).values.size() > 0 ){
        // returning the last available value...
        return programSettingsVector.at(i).values.at( programSettingsVector.at(i).values.size()-1 );
      }
      else {msgOut(MSG_CRITICAL_ERROR, "Error: "+name_h+" doesn’t have any value, even on the first position(year)!"); }
    }
  }
  if(type_h==TYPE_BOOL){
    msgOut(MSG_DEBUG, "Possible error calling getBaseData(TYPE_BOOL) for "+ name_h +". No setting option or macro data found with this name. Returning false.");
    return "0";
  } else {
    msgOut(MSG_CRITICAL_ERROR, "Error calling getBaseData() for "+ name_h +". No setting option or macro data found with this name.");
  }
  return "";
}

vector <string>
ModelData::getVectorBaseData (const string &name_h, int type_h){
  for (uint i=0; i<programSettingsVector.size();i++){
    if (programSettingsVector.at(i).name == name_h){
      int type = programSettingsVector.at(i).type;
      if(type != type_h){msgOut(MSG_CRITICAL_ERROR, "mismatching type in calling getVectorBaseData() for "+name_h);}
      return programSettingsVector.at(i).values;
    }
  }
  msgOut(MSG_CRITICAL_ERROR, "Error calling getVectorBaseData() for "+ name_h +". No setting option or macro data found with this name.");
  vector <string> toReturn;
  return toReturn;
}

// ------------ start getSetting() amd getMacro() functions ....... -----------------
int
ModelData::getIntSetting(const string &name_h, int position) const{
  return s2i( MTHREAD->MD->getBaseData(name_h,TYPE_INT,position) );
}
double
ModelData::getDoubleSetting(const string &name_h, int position) const{
  return s2d( MTHREAD->MD->getBaseData(name_h,TYPE_DOUBLE,position) );
}
string
ModelData::getStringSetting(const string &name_h, int position) const{
  return MTHREAD->MD->getBaseData(name_h,TYPE_STRING,position);
```

```
}
bool
ModelData::getBoolSetting(const string &name_h, int position) const{
  return s2b( MTHREAD->MD->getBaseData(name_h,TYPE_BOOL,position) );
}
vector<int>
ModelData::getIntVectorSetting(const string &name_h) const{
  return s2i(MTHREAD->MD->getVectorBaseData(name_h, TYPE_INT));
}
vector<double>
ModelData::getDoubleVectorSetting(const string &name_h) const{
  return s2d(MTHREAD->MD->getVectorBaseData(name_h, TYPE_DOUBLE));
}
vector<string>
ModelData::getStringVectorSetting(const string &name_h) const{
  return MTHREAD->MD->getVectorBaseData(name_h, TYPE_STRING);
}
vector<bool>
ModelData::getBoolVectorSetting(const string &name_h) const{
  return s2b(MTHREAD->MD->getVectorBaseData(name_h, TYPE_BOOL));
}

// ------ END of getSetting() functions ---------------------

void
ModelData::setBasicData(const string &name_h, int value, int position){
  setBasicData(name_h, i2s(value), TYPE_INT, position);
}
void
ModelData::setBasicData(const string &name_h, double value, int position){
  setBasicData(name_h, d2s(value), TYPE_DOUBLE, position);
}
void
ModelData::setBasicData(const string &name_h, string value, int position){
  setBasicData(name_h, value, TYPE_STRING, position);
}
void
ModelData::setBasicData(const string &name_h, bool value, int position){
  setBasicData(name_h, b2s(value), TYPE_BOOL, position);
}

void
ModelData::setBasicData(const string &name_h, string value, int type_h, int position){
  for (uint i=0; i<programSettingsVector.size();i++){
    if (programSettingsVector.at(i).name == name_h){
      int type = programSettingsVector.at(i).type;
      if(type != type_h){msgOut(MSG_CRITICAL_ERROR, "mismatching type in calling setBasicData() for "+name_h);}
      if(programSettingsVector.at(i).values.size() > ((uint)position)) {
        programSettingsVector.at(i).values.at(position)=value;
        return;
      }
      else {msgOut(MSG_CRITICAL_ERROR, "out-of-bound error calling setBasicData() for "+name_h); }
    }
  }
  msgOut(MSG_CRITICAL_ERROR, "Error calling setBasicData() for "+ name_h +". No setting option or macro data found with this name.");
  return;
}

std::string
ModelData::getFilenameByType(std::string type_h){
  std::string directory;
  std::string filename;
  std::string filename_complete;
  for (uint i=0; i<iFilesVector.size(); i++){
    if (iFilesVector.at(i).type == type_h){
      directory=iFilesVector.at(i).directory;
      filename=iFilesVector.at(i).name;
      break;
    }
  }
  filename_complete = baseDirectory+directory+filename;
  return filename_complete;
}

vector <string>
ModelData::getDiameterClasses(bool productionOnly){
  int i;
  if(productionOnly){
    i=1;
  } else {
    i=0;
  }
  vector <string> toReturn;
  for (i;i<diamClasses.size();i++){
    toReturn.push_back(diamClasses[i]);
  }
  return toReturn;
}

/**
Basic function to retrieve products-related data.
It addmits the following "filters":
@type_h Name of the specific parameter requested
@regId_h Look for level1 or level 2 region.
@prodId_h Product. It accept three keywords, for summing up all products, primary products or secondary products, namelly PROD_ALL, PROD_PRI, PROD_SEC.
@year Unless specified, get the value of the current year. If array is smaller (e.g. because it is time-independent), get the last value.
@freeDim_h If specified, look exactly for it, otherwise simply doesn't filter for it.

*/
const double
ModelData::getProdData(const string &type_h, const int& regId_h, const string & prodId_h, const int& year, const string &freeDim_h) {

  double value=0;
  vector <int> regIds;
  string key;
  DataMap::const_iterator p;

  bool found = false;
  vector <string> products;
  bool exactMatch=true;

  if(prodId_h == PROD_PRI){
    products = priProducts;
  } else if (prodId_h == PROD_SEC){
    products = secProducts;
  } else if (prodId_h == PROD_ALL || prodId_h == ""){
    products = allProducts;
    products.push_back("");
  } else  {
    products.push_back(prodId_h);
  }
  if(freeDim_h=="") exactMatch=false;

  // Make sure to set the new value to all l2 regions if requested for a reg1 level
  if(getRegion(regId_h)->getRegLevel()==2){
    regIds.push_back(regId_h);
  } else if (getRegion(regId_h)->getRegLevel()==1) {
    for(uint i=0;i<getRegion(regId_h)->getNChildren();i++){
      regIds.push_back(getRegion(regId_h)->getChildren()[i]->getRegId());
    }
  } else {
    msgOut(MSG_CRITICAL_ERROR, "Error in setProdData(). Setting a value for the whole World is not supported.");
  }
  int regIdsS = regIds.size();


  for(uint r=0;r<regIdsS;r++){
    for(uint i=0;i<products.size();i++){
      key = makeKeyProdData(type_h,i2s(regIds[r]),products[i],freeDim_h);
      if (!exactMatch && key.size () > 0)  key.resize (key.size () - 1); // bug 20140402, removing the last #
      value += dataMapGetValue(prodDataMap,key,year,exactMatch);
      if(tempBool) found = true;
    }
  }

  if(!found){
    msgOut(errorLevel, "Error in getProdData: no combination found for "+type_h+", "+i2s(regId_h)+", "+prodId_h+", "+i2s(year)+", "+freeDim_h+". Returning 0, but double check that this is ok for your model.");
  }
  return value;


}

/**
Basic function to retrieve forest-related data.
It addmits the following "filters":
@type_h Name of the specific parameter requested
@regId_h Look for a level1 or level2 region
@forType_h If specified, look exactly for the specified forest type, otherwise accept the keyword FT_ALL
```

```
 for summing all of them
@freeDim_h Normally used for diameter class, but occasionally used for other uses (changed 20140514). It
 accepts three keywords, for summing up all diameters, production-ready diameters or sub-production ones,
 namelly DIAM_ALL, DIAM_PROD, DIAM_FIRST.\\
If a diameter-independed variable is required, put it in the data with an empty diameter class and retrieve
 it here using DIAM_ALL.
@year Unless specified, get the value of the current year. If array is smaller (e.g. because it is
 time-independent), get the last value.
*/
const double
ModelData::getForData(const string &type_h, const int& regId_h, const string &
forType_h, const string &freeDim_h, const int& year){
  vector<int> regIds;
  vector <string> dClasses;
  vector <string> fTypes;
  string key;
  DataMap::const_iterator p;
  bool found = false;
  double value = 0;

  // creating the arrays to look up if keywords were specified..
  if (forType_h == FT_ALL){ // || forType_h == ""){
    fTypes = getForTypeIds();
    fTypes.push_back("");
  } else {
    fTypes.push_back(forType_h);
  }
  if(freeDim_h == DIAM_ALL){ // || freeDim_h == ""){
    dClasses = diamClasses;
    dClasses.push_back("");
  } else if (freeDim_h == DIAM_PROD){
    dClasses = getDiameterClasses(true);
  } else if (freeDim_h == DIAM_FIRST){
    dClasses.push_back(diamClasses.at(0));
  } else  {
    dClasses.push_back(freeDim_h);
  }
  // Make sure to set the new value to all l2 regions if requested for a reg1 level
  if(getRegion(regId_h)->getRegLevel()==2){
    regIds.push_back(regId_h);
  } else if (getRegion(regId_h)->getRegLevel()==1) {
    for(uint i=0;i<getRegion(regId_h)->getNChildren();i++){
      regIds.push_back(getRegion(regId_h)->getChildren()[i]->getRegId());
    }
  } else {
    msgOut(MSG_CRITICAL_ERROR, "Error in getProdData(). Setting a value for the
 whole World is not supported.");
  }
  int regIdsS = regIds.size();

  // getting the actual data...
  for(uint r=0;r< regIds.size();r++){
    for(uint i=0;i<dClasses.size();i++){
      for (uint y=0;y<fTypes.size();y++){
        key = makeKeyForData(type_h,i2s(regIds[r]),fTypes[y],dClasses[i]);
        value += dataMapGetValue(forDataMap,key,year,true);
        if(tempBool) found = true;
      }
    }
  }

  if(!found){
        msgOut(errorLevel, "Error in getForData(): no combination found for "+type_h+", "+
i2s(regId_h)+", "+forType_h+", "+i2s(year)+", "+freeDim_h+". Returning 0, but double check that this
 is ok for your model.");
  }
  return value;
}


/**
Basic function to set products-related data.
It can change an existing value or extend in time a serie, but it requires the keys (par. name/regId/prodId
/freedim) to be already present in the data.
@value_h New value to change with/add
It addmits the following "filters":
@type_h Name of the specific parameter requested
@regId_h Set a specific level 2 region, or all its childred l2 region if a reg1 level is specified.
@prodId_h Product. It accept three keywords, for changing/inserting the new value to all products, primary
 products or secondary products, namelly PROD_ALL, PROD_PRI, PROD_SEC.
@year Unless specified, set the value of the current year. If array is smaller (e.g. because it is
 time-independent) fill all the values till the requested one.
@create If true, allow creation of new data if not found. Default false (rise an error)
@freeDim_h If specified, look exactly for it, otherwise simply doesn't filter for it.

*/
void
```

```
ModelData::setProdData(const double& value_h, const string &type_h, const int&
regId_h, const string &prodId_h, const int& year, const bool& allowCreate, const string &freeDim_h){

  vector<int> regIds;
  string key;
  DataMap::const_iterator p;
  vector <string> products;

  if(prodId_h == PROD_PRI){
    products = priProducts;
  } else if (prodId_h == PROD_SEC){
    products = secProducts;
  } else if (prodId_h == PROD_ALL){
    products = allProducts;
  } else  {
    products.push_back(prodId_h);
  }

  // Make sure to set the new value to all l2 regions if requested fora reg1 level
  if(getRegion(regId_h)->getRegLevel()==2){
    regIds.push_back(regId_h);
  } else if (getRegion(regId_h)->getRegLevel()==1) {
    for(uint i=0;i<getRegion(regId_h)->getNChildren();i++){
      regIds.push_back(getRegion(regId_h)->getChildren()[i]->getRegId());
    }
  } else {
    msgOut(MSG_CRITICAL_ERROR, "Error in setProdData(). Setting a value for the
 whole World is not supported.");
  }

  bool found = false;
  bool tempFound = false;

  for(uint r=0;r< regIds.size();r++){
    for(uint i=0;i<products.size();i++){
      key = makeKeyProdData(type_h,i2s(regIds[r]),products[i],freeDim_h);
      tempFound = dataMapSetValue(prodDataMap,key,value_h, year,true);
      if(tempFound) found = true;
    }
  }

  if(!found){
    if(!allowCreate){
      msgOut(MSG_CRITICAL_ERROR, "Error in setProdData: no combination found for "+
type_h+", "+i2s(regId_h)+", "+prodId_h+", "+i2s(year)+", "+freeDim_h+". You can allow new variables to
 be created using the \"allowCreate\" flag.");
    } else {
      for(uint r=0;r< regIds.size();r++){
        for(uint i=0;i<products.size();i++){
          key = makeKeyProdData(type_h,i2s(regIds[r]),products[i],freeDim_h);
          vector <double> values;
          setTimedData(value_h,values,year,MSG_NO_MSG);
          prodDataMap.insert(DataPair(key,values));
        }
      }
    }
  }

}




void
ModelData::setForData(const double& value_h, const string &type_h, const int& regId_h,
 const string &forType_h, const string &freeDim_h, const int& year, const bool& allowCreate){

  vector<int> regIds;
  vector <string> dClasses;
  vector <string> fTypes;
  string key;
  DataMap::const_iterator p;
  bool found = false;
  bool tempFound = false;

  if (forType_h == FT_ALL){
    fTypes = getForTypeIds();
  } else {
    fTypes.push_back(forType_h);
  }

    if(freeDim_h == DIAM_ALL){
    dClasses = diamClasses;
    } else if (freeDim_h == DIAM_PROD){
    dClasses = getDiameterClasses(true);
    } else if (freeDim_h == DIAM_FIRST){
    dClasses.push_back(diamClasses.at(0));
  } else  {
        dClasses.push_back(freeDim_h);
  }

  // Make sure to set the new value to all l2 regions if requested for a reg1 level
  if(getRegion(regId_h)->getRegLevel()==2){
    regIds.push_back(regId_h);
  } else if (getRegion(regId_h)->getRegLevel()==1) {
    for(uint i=0;i<getRegion(regId_h)->getNChildren();i++){
      regIds.push_back(getRegion(regId_h)->getChildren()[i]->getRegId());
    }
  } else {
    msgOut(MSG_CRITICAL_ERROR, "Error in setProdData(). Setting a value for the whole World is not supported.");
  }
  int regIdsS = regIds.size();

  for(uint r=0;r< regIds.size();r++){
    for(uint i=0;i<dClasses.size();i++){
      for (uint y=0;y<fTypes.size();y++){
        key = makeKeyForData(type_h,i2s(regIds[r]),fTypes[y],dClasses[i]);
        tempFound = dataMapSetValue(forDataMap,key,value_h, year,true);
        if(tempFound) found = true;
      }
    }
  }

  if(!found){
    if(!allowCreate){
            msgOut(MSG_CRITICAL_ERROR, "Error in setForData: no combination found for "+type_h+", "+i2s(regId_h)+", "+forType_h+", "+i2s(year)+", "+freeDim_h+". You can allow new variables to be created using the \"allowCreate\" flag.");
    } else {
      for(uint r=0;r< regIds.size();r++){
        for(uint i=0;i<dClasses.size();i++){
          for (uint y=0;y<fTypes.size();y++){
            key = makeKeyForData(type_h,i2s(regIds[r]),fTypes[y],dClasses[i]);
            vector <double> values;
            setTimedData(value_h,values,year,MSG_NO_MSG);
            forDataMap.insert(DataPair(key,values));
          }
        }
      }
    }
  }
}


double
ModelData::getTimedData(const vector<double> &dated_vector, const int& year_h) const
{

  int position;
  if(year_h==DATA_NOW){
    position = MTHREAD->SCD->getYear()-cached_initialYear;
  } else {
    position = year_h-cached_initialYear;
  }

  if(dated_vector.size() > position) {
    return dated_vector[position];
  } else if (dated_vector.size() > 0 ){
    // returning the last available value...
    return dated_vector[dated_vector.size()-1];
  } else {
    msgOut(MSG_CRITICAL_ERROR, "Error in getTimedData: requested value doesn't have any value, even on the first position(year)!");
  }
  return 0;
}

void
ModelData::setTimedData(const double& value_h, vector<double> &dated_vector, const int& year_h, const int& MSG_LEVEL){

  int position;
  if(year_h==DATA_NOW){
    position = MTHREAD->SCD->getYear()-cached_initialYear;
  } else {
    position = year_h-cached_initialYear;
  }

  int originalVectorSize = dated_vector.size();
  if(dated_vector.size() > position) {
    dated_vector[position]=value_h;
  } else {
    // extending the vector and filling it with the incoming value, but issuing a warning if done for more
```

```
 than one year

    for(uint i=0;i<position-originalVectorSize+1;i++){
      dated_vector.push_back(value_h);
    }
    if(position-originalVectorSize > 0 ){
      msgOut(MSG_LEVEL, "setTimedData: a dated vector has been filled several years ("+
i2s(1+position-originalVectorSize)+") with incoming values to reach desidered position in time.");
    }
  }
}


void
ModelData::loadInput(){
  msgOut(MSG_INFO, "Loading input files (this can take a few minutes)...");
  //QString iFile("data/ffsmInput.ods");
  QString iFile(MTHREAD->getInputFileName().c_str());
  //cout << "PIPPO !!!!! " << MTHREAD->getInputFileName().c_str() << endl;

  //std::random_device rd;
  //std::mt19937 localgen(rd());
  std::mt19937 localgen(time(0));
  std::uniform_int_distribution<> dis(10, 1000000);
  int randomNumber = dis(localgen);

  QString oDir((MTHREAD->getBaseDirectory()+"tempInput-"+
MTHREAD->getScenarioName()+i2s(randomNumber)).c_str());
  string forDataCachedFilename = MTHREAD->getBaseDirectory()+"
cachedInput/forData.csv";
  string prodDataCachedFilename = MTHREAD->getBaseDirectory()+"
cachedInput/prodData.csv";

  // removing output directory if exist..
  QDir oQtDir(oDir);

  if(oQtDir.exists()){
    bool deleted;
    deleted = delDir(oDir);
    if(deleted){msgOut(MSG_DEBUG,"Correctly deleted old temporary data");}
    else {msgOut(MSG_WARNING, "I could not delete old temporary data dir (hopefully we’ll
 overrite the input files)");}
  }

  if (!QFile::exists(iFile))
  {
    cout << "File does not exist." << endl << endl;
    //return false;
  }
  UnZip::ErrorCode ec;
  UnZip uz;
  ec = uz.openArchive(iFile);
  if (ec != UnZip::Ok)  {
        //cout << "Failed to open archive: " << uz.formatError(ec).toAscii().data() << endl << endl; // Qt4
        cout << "Failed to open archive: " << uz.formatError(ec).toLatin1().data() << endl <<
endl;  // Qt5
    //return false;
  }
  ec = uz.extractAll(oDir);
  if (ec != UnZip::Ok){
    //cout << "Extraction failed: " << uz.formatError(ec).toAscii().data() << endl << endl; // Qt4
    cout << "Extraction failed: " << uz.formatError(ec).toLatin1().data() << endl << endl;  //
 Qt5
    uz.closeArchive();
    //return false;
  }

  // loading input file into memory...
  string inputXMLFileName = MTHREAD->getBaseDirectory()+"tempInput-"+
MTHREAD->getScenarioName()+i2s(randomNumber)+"/content.xml";
  //string inputXMLFileName = MTHREAD->getBaseDirectory()+"test/content.xml";
  //cout << "inputXMLFileName: " << inputXMLFileName << endl;
  //mainDocument = new InputDocument();
  mainDocument.setWorkingFile(inputXMLFileName);
  //InputNode documentContent = mainDocument.getNodeByName("office:document-content");
  //InputNode documentBody = mainDocument.getNodeByName("office:body");
  //InputNode mainNode = mainDocument.getNodeByName("spreadsheet");
  //InputNode pippo = mainDocument.getNodeByName("pippo-pippo");
  //InputNode table = mainDocument.getNodeByName("table");
  //cout << "Test result: " << table.getStringContent() << endl;


  vector <InputNode> tables = mainDocument.getNodesByName("table");
  for(uint i=0;i<tables.size();i++){
    string tableName = tables[i].getStringAttributeByName("name");
    //cout <<tableName<<endl;
    if( tableName == "forData"){
```

```
    if(QFile::exists(forDataCachedFilename.c_str())){
     loadDataFromCache("forData");
     continue;
    }
   } else if (tableName == "prodData"){
     if (QFile::exists(prodDataCachedFilename.c_str())) {
      loadDataFromCache("prodData");
     continue;
     }
   }
   LLData data(MTHREAD,tables[i].getStringAttributeByName("name"));
   vector <InputNode> rows = tables[i].getNodesByName("table-row",MSG_NO_MSG,true);
   if(rows.size()<2) continue; //empty table or only with headers
   // building headers..
   vector <InputNode> cells = rows[0].getNodesByName("table-cell",MSG_NO_MSG,true);
   for (uint y=0; y<cells.size(); y++){
    int repeated = 1;
    if( cells[y].hasAttributeByName("number-columns-repeated")){
     repeated = cells[y].getIntAttributeByName("number-columns-repeated");
    }
    for (int q=0;q<repeated;q++){
     if( !cells[y].hasChildNode("p") ){
      data.headers.push_back(""); // empty header
     } else {
      data.headers.push_back(cells[y].getNodeByName("p",MSG_NO_MSG,true).
getStringContent());
     }
    }
   }
   // loading data...
   for (uint j=1; j<rows.size();j++){
    //cout << j << endl;
    vector <InputNode> cells = rows[j].getNodesByName("table-cell",MSG_NO_MSG,true);
    //vector <InputNode> cells = rows[j].getChildNodes();
    if (cells.size()<1) continue;
    vector<string> record;
    // checking the first cell is not a comment nor is empty..
    int childCount = cells[0].getChildNodesCount();
    if (childCount == 0 || !cells[0].hasChildNode("p")) continue; // empty line, first column empty!
    string fistCol = cells[0].getNodeByName("p",MSG_NO_MSG,true).getStringContent();
    unsigned int z;
    z = fistCol.find("#");
    if( z!=string::npos && z == 0) continue; // found "#" on fist position, it's a comment!
    for (uint y=0; y<cells.size(); y++){
     int repeated = 1;
     if( cells[y].hasAttributeByName("number-columns-repeated")){
      repeated = cells[y].getIntAttributeByName("number-columns-repeated");
     }
     for (int q=0;q<repeated;q++){
      if( !cells[y].hasChildNode("p") ){
       record.push_back(""); // empty header
      } else {
       // changed 20120625 as for float values the content of p is the visualised value, not the full
 memorised one.
       // this is strange because  tought I already tested it.. but maybe is changed the format??
       if(cells[y].getStringAttributeByName("value-type")=="float"){
        record.push_back(cells[y].getStringAttributeByName("value"));
       } else {
        record.push_back(cells[y].getNodeByName("p",MSG_NO_MSG,true).getStringContent());
       }
      }
     }
    }
    data.records.push_back(record);
   }
   data.clean();
   LLDataVector.push_back(data);
  }

  //debug !!!
  /*for (uint i=0; i<LLDataVector.size();i++){
   cout << "***************** NEW TABLE: " << LLDataVector[i].tableName << endl;
   //cout << "***** Headers: "<< endl;
   int headerSize = LLDataVector[i].headers.size();
   bool ok = true;
   cout << "Header size: " << headerSize << endl;
   //for (uint j=0; j<LLDataVector[i].headers.size();j++){
   //  cout << "["<<j<<"] " << LLDataVector[i].headers[j] << endl;
   //}
   //cout << "***** Records: " << endl;
   for (uint j=0; j<LLDataVector[i].records.size();j++){
    //cout << "** Record "<<j<<":"<<endl;
    if(LLDataVector[i].records[j].size() != headerSize){
     cout << "There is a problem on record " << j <<"!"<< endl;
     cout << "His size is: "<< LLDataVector[i].records[j].size() << endl;
     ok = false;
    }
```

```
      //for (uint y=0; y<LLDataVector[i].records[j].size();y++){
      //  cout << "["<<y<<"] " << LLDataVector[i].records[j][y] << endl;
      //}
    }
    if(!ok) {cout <<"Problems with this table :-( !"<<endl;}
  }*/



  // deleting output directory if exist...
  if(oQtDir.exists()){
    bool deleted;
    deleted = delDir(oDir);
    if(deleted){msgOut(MSG_DEBUG,"Correctly deleted old temporary data");}
    else {msgOut(MSG_WARNING, "I could not delete old temporary data dir (hopefully we’ll overrite the input files)");}
  }
}



void
ModelData::loadDataFromCache(string tablename){
  msgOut(MSG_INFO,"Attenction, using cached data (csv) for "+tablename);
  string fileName = MTHREAD->getBaseDirectory()+"cachedInput/"+tablename+".csv";
  QFile file(fileName.c_str());
  if (!file.open(QFile::ReadOnly)) {
      msgOut(MSG_ERROR, "Cannot open cached file "+fileName+" for reading.");
  }
  QTextStream in(&file);
  LLData data(MTHREAD, tablename);
  int countRow = 0;
  while (!in.atEnd()) {
    QString line = in.readLine();
    QStringList fields = line.split(’;’);
    if (countRow==0){ // building headers
      for(uint i =0;i<fields.size();i++){
        data.headers.push_back(fields.at(i).toStdString());
      }
    } else {
      vector<string> record ; //= fields.toVector().toStdVector();
      unsigned int z = fields[0].toStdString().find("#");
      if( z!=string::npos && z == 0) continue; // found "#" on fist position, it’s a comment!
      for(uint i =0;i<fields.size();i++){
        string field = fields.at(i).toStdString();
        replace(field.begin(), field.end(), ’,’, ’.’);
        record.push_back(field);
      }
      data.records.push_back(record);
    }
    countRow++;
  }
  data.clean();
  LLDataVector.push_back(data);

}

bool
ModelData::delDir(QString dirname) {
        bool deleted = false;
        QDir dir(dirname);
  //msgOut(MSG_DEBUG, dir.absolutePath().toStdString());
  dir.setFilter(QDir::Dirs | QDir::Files | QDir::NoDotAndDotDot | QDir::NoSymLinks);
  QFileInfoList list = dir.entryInfoList();
  deleted = dir.rmdir(dir.absolutePath());
  if (deleted) return true;

  for (int i = 0; i < list.size(); ++i) {
    QFileInfo fileInfo = list.at(i);
    if (fileInfo.isFile()){
      //msgOut(MSG_DEBUG, "A file, gonna remove it: "+fileInfo.absoluteFilePath().toStdString());
      QFile targetFile(fileInfo.absoluteFilePath());
      bool fileDeleted = targetFile.remove();
      if (!fileDeleted){
        msgOut(MSG_CRITICAL_ERROR, "We have a problem: can’t delete file "+fileInfo.absoluteFilePath().toStdString());
      }
    }
    else if (fileInfo.isDir()){
      //msgOut(MSG_DEBUG, "A directory, gonna go inside it: "+fileInfo.absoluteFilePath().toStdString());
      delDir(fileInfo.absoluteFilePath());
      dir.rmdir(fileInfo.absoluteFilePath());
    }
  }

  deleted = dir.rmdir(dir.absolutePath());
  if (deleted) return true;
  return false;
}


LLData
ModelData::getTable(string tableName_h, int debugLevel){
  LLData toReturn(MTHREAD,"");
  for(uint i=0;i<LLDataVector.size();i++){
    if (LLDataVector[i].getTableName() == tableName_h)return
LLDataVector[i];
  }
  msgOut(debugLevel,"No table found with name "+tableName_h);
  return toReturn;
}


bool
ModelData::dataMapCheckExist(const DataMap& map, const string&
search_for, const bool& exactMatch) const {
  /*int dummyYear=MTHREAD->SCD->getYear();
  if(dataMapGetValue(map, search_for, dummyYear, exactMatch)==DATA_ERROR) {
    return false;
  } else {
    return true;
  }
  return false;
}*/
  bool found = false;
  DataMap::const_iterator i;
  if(!exactMatch){
    i = map.lower_bound(search_for);
    for(;i != map.end();i++){
      const string& key = i->first;
      if (key.compare(0, search_for.size(), search_for) == 0) {// Really a prefix?
          return true;
      } else {
        return false;
      }
    }
  } else {
    i = map.find(search_for);
    if (i!=map.end()){
      return true;
    }
  }
  return false;
}


double
ModelData::dataMapGetValue(const DataMap& map, const string& search_for,
const int& year_h, const bool& exactMatch) {
  double toReturn = 0;
  tempBool = false;
  DataMap::const_iterator i;
  if(!exactMatch){
    i = map.lower_bound(search_for);
    for(;i != map.end();i++){
      const string& key = i->first;
      if (key.compare(0, search_for.size(), search_for) == 0) {// Really a prefix?
          tempBool = true;
          toReturn += getTimedData( i->second, year_h );
      } else {
        break;
      }
    }
  } else {
    i = map.find(search_for);
    if (i!=map.end()){
      tempBool = true;
      return  getTimedData( i->second, year_h );
    }
  }
  return toReturn;
}



int
ModelData::dataMapSetValue( DataMap& map, const string& search_for, const
double& value_h, const int& year_h, const bool& exactMatch){
  bool found = false;
  DataMap::iterator i;
  if(!exactMatch){
    i = map.lower_bound(search_for);
    for(;i != map.end();i++){
      const string& key = i->first;
      if (key.compare(0, search_for.size(), search_for) == 0) {// Really a prefix?
```

```
          found = true;
          setTimedData(value_h, i->second, year_h);
      } else {
        break;
      }
    }
  } else {
    i = map.find(search_for);
    if (i!=map.end()){
      found = true;
      setTimedData(value_h, i->second, year_h, errorLevel);
    }
  }
  // removed 20120903 as the insertion of new values must be explicitly done, not in all cases we want a new insertion
  /*if(!found){
    vector < double> newValues;
    setTimedData(value_h, newValues, year_h, MSG_NO_MSG); // don't warning if we are making a multi-year value vector, as it is a new one
    map.insert(DataPair (search_for,newValues));
  }*/
  return found;
}

void
ModelData::unpackKeyProdData(const string& key, string& parName, int& regId, string& prod, string& freeDim) const{

  int parNameDelimiter = key.find("#",0);
  int regIdDelimiter = key.find("#",parNameDelimiter+1);
  int prodDelimiter = key.find("#",regIdDelimiter+1);
  int freeDimDelimiter = key.find("#",prodDelimiter+1);
  if (freeDimDelimiter == string::npos){
    msgOut(MSG_CRITICAL_ERROR, "Error in unpacking a key in the map of production data.");
  }
  parName.assign(key,0,parNameDelimiter);
  string regIdString="";
  regIdString.assign(key,parNameDelimiter+1,regIdDelimiter-parNameDelimiter-1);
  regId = s2i(regIdString);
  prod.assign(key,regIdDelimiter+1,prodDelimiter-regIdDelimiter-1);
  freeDim.assign(key,prodDelimiter+1,freeDimDelimiter-prodDelimiter-1);

}

void
ModelData::unpackKeyForData(const string& key, string& parName, int &regId, string& forType, string& diamClass) const{
  int parNameDelimiter = key.find("#",0);
  int regIdDelimiter = key.find("#",parNameDelimiter+1);
  int forTypeDelimiter = key.find("#",regIdDelimiter+1);
  int diamClassDelimiter = key.find("#",forTypeDelimiter+1);
  if (diamClassDelimiter == string::npos){
    msgOut(MSG_CRITICAL_ERROR, "Error in unpacking a key in the map of production data.");
  }
  parName.assign(key,0,parNameDelimiter);
  string regIdString="";
  regIdString.assign(key,parNameDelimiter+1,regIdDelimiter-parNameDelimiter-1);
  regId = s2i(regIdString);
  forType.assign(key,regIdDelimiter+1,forTypeDelimiter-regIdDelimiter-1);
  diamClass.assign(key,forTypeDelimiter+1,diamClassDelimiter-forTypeDelimiter-1);

}


/**
calculating the discount factor

Revenues at years n will be transforemed as average year rate as

av.y.rev = rev(n)/ ( (1+ir)^(n-1)+(1+ir)^(n-2)+(1+ir)^(n-3)+...+(1+ir)^(n-n) )

Objective is to have the present value of the final harvest (A) equal to the sum pf the present values of yearly activities (B):

\image html diagram_calculateAnnualisedEquivalent.png "Comparing present values" width=10cm

\f[ PV(A) = SUM(PV(B) \f]
\f[ A/(1+r)^n = B/(1+r)^1 + B/(1+r)^2 + ... + B/(1+r)^n \f]
\f[ A/(1+r)^n = B * ( 1/(1+r)^1 + 1/(1+r)^2 + ... + 1/(1+r)^n ) \f]
\f[ A/(1+r)^n = B * ( (1+r)^(n-1) + (1+r)^(n-2) + ... + (1+r)^(n-n) ) \f]
\f[ B = A / ( (1+r)^(n-1) + (1+r)^(n-2) + ... + (1+r)^(n-n) ) \f]

20131204. Changed for the equivalent but simpler eai = rev(t)*i / ((1+i)^t-1)

*/
```

```
double
ModelData::calculateAnnualisedEquivalent(double amount_h, int years_h){
  // modified and tested 20120912. Before it was running this formula instead:
  // av.y.rev = rev(n)/ ( (1+ir)^1+(1+ir)^2+(1+ir)^3+...+(1+ir)^n )
  // the difference is that in this way the annual equivalent that is calulated doesn't need to be further discounted for yearly activites (e.g. agric)

  //loop(fy$(ord(fy)=1),
  //  df(fy)= (1+ir)**(ord(fy));
  //);
  //loop(fy$(ord(fy)>1),
  //  df(fy)=df(fy-1)+(1+ir)**(ord(fy));
  //);
  if(years_h<0) return 0.;
  if(years_h==0) return amount_h;
  double ir = getDoubleSetting("ir");
  double eai = amount_h * ir / (pow(1.0+ir,years_h)-1.0);

  return eai;

    /*
  vector <double> df_by;
  for(int y=0;y<years_h;y++){
    double df;
    if(y==0){
      df = pow((1+ir),y);
    } else {
      df = df_by.at(y-1)+pow((1+ir),y);
    }
    if (y==years_h-1) {
         cout << eai << "   " << amount_h/df << endl;
     return amount_h/df; // big bug 20120904
    }
    df_by.push_back(df);
  }
  exit(1);
  return 0; // never reached, just to avoid compilation warnings
    */
}

double
ModelData::calculateAnnualisedEquivalent(double amount_h, double years_h){
  //ceil(x) DNLP returns the smallest integer number greater than or equal to x
  //loop( (u,i,lambda,essence),
  //  cumTp(u,i,lambda,essence) =   ceil(cumTp(u,i,lambda,essence));
  //);
  int ceiledYear = ceil(years_h);
  return calculateAnnualisedEquivalent(amount_h, ceiledYear);
}

/** Get a list of files in a directory */
int
ModelData::getFilenamesByDir (const string & dir, vector<string> &files, const string & filter){
  DIR *dp;
  struct dirent *dirp;
  if((dp  = opendir(dir.c_str())) == NULL) {
    msgOut(MSG_ERROR, "Error " + i2s(errno) + " opening the " + dir + " directory.");
    //cout << "Error(" << errno << ") opening " << dir << endl;
    return errno;
  }
  while ((dirp = readdir(dp)) != NULL) {
    string filename = dirp->d_name;
    if(
      (filter != "" && filename.substr(filename.find_last_of(".")) == filter) // there is a filter and the last bit of the filename match the filter
      || (filter == "" && filename.substr(filename.find_last_of(".") + 1) != "") // there isn't any filter but we don't want stuff like ".." or "."
    ) {
      files.push_back(string(dirp->d_name));
    }
  }
  closedir(dp);
  return 0;
}


vector<pathRule*>
ModelData::getPathMortalityRule(const string& forType, const string& dC){
  vector<pathRule*> toReturn;
  for(uint i=0;i<pathRules.size();i++){
    if(pathRules[i].forType == forType && pathRules[i].dClass == dC){
      toReturn.push_back(&pathRules[i]);
    }
  }
  return toReturn;
}

/**
 * @brief ModelData::createCombinationsVector
 * Return a vector containing any possible combination of nItems items (including all subsets).
 *
 * For example with nItems = 3:
 * 0: []; 1: [0]; 2: [1]; 3: [0,1]; 4: [2]; 5: [0,2]; 6: [1,2]; 7: [0,1,2]

 * @param nItems number of items to create p
 * @return A vector with in each slot the items present in that specific combination subset.
 */
vector < vector <int> >
ModelData::createCombinationsVector(const int& nItems) {
  // Not confuse combination with permutation where order matter. Here it doesn't matter, as much as the algorithm is the same and returns
  // to as each position always the same subset
  vector < vector <int> > toReturn;
  int nCombs = pow(2,nItems);
  //int nCombs = nItems;
  for (uint i=0; i<nCombs; i++){
    vector<int> thisCombItems; //concernedPriProducts;
    for(uint j=0;j<nItems;j++){
      uint j2 = pow(2,j);
      if(i & j2){ // bit a bit operator, p217 C++ book
        thisCombItems.push_back(j);
      }
    }
    toReturn.push_back(thisCombItems);
  }
  return toReturn;
}


double
ModelData::getAvailableDeathTimber(const vector<string> &primProd_h, int regId_h, int year_h){
  if (!getBoolSetting("useDeathTimber")) return 0;
  double toReturn = 0.0;
  vector <string> forTypesIds = getForTypeIds();
  for (uint i=0;i<forTypesIds.size();i++){
    string ft = forTypesIds[i];
    for(uint u=0;u<diamClasses.size();u++){
      string dc = diamClasses[u];
      bool possible = false;
      int maxYears = 0;
      for (int p=0; p<primProd_h.size();p++){
          string primProd = primProd_h[p];
          if(assessProdPossibility(primProd,ft, dc)){
              possible = true;
              maxYears=max(maxYears,getMaxYearUsableDeathTimber(primProd, ft, dc));
          }
      }
      if(possible){
        for(int y=year_h;y>year_h-maxYears;y--){
          iisskey key(y,regId_h,ft,dc);
          toReturn += findMap(deathTimberInventory,key,MSG_DEBUG,0.0);
        }
      }
    }
  }
  return toReturn;
}

vector <int>
ModelData::getAllocableProductIdsFromDeathTimber(const int &regId_h, const string &ft, const string &dc, const int &harvesting_year, int request_year){
  vector<int> allocableProductIds;
  if (!getBoolSetting("useDeathTimber")) return allocableProductIds;
  if (request_year == DATA_NOW) request_year = MTHREAD->SCD->getYear();
  for(uint p=0;p<priProducts.size();p++){
    string primProd = priProducts[p];
    if(assessProdPossibility(primProd,ft, dc)){
      int maxYears = getMaxYearUsableDeathTimber(primProd, ft, dc);
      if (request_year-harvesting_year < maxYears){
        allocableProductIds.push_back(p);
      }
    }
  }
  return allocableProductIds;
}

```

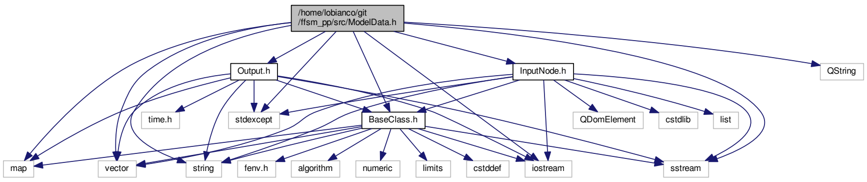
```


double
ModelData::getAvailableAliveTimber(const vector<string> &primProd_h, int regId_h){
  double toReturn = 0.0;
  ModelRegion* REG = MTHREAD->MD->getRegion(regId_h);
  vector <Pixel*> regPx = REG->getMyPixels();
  vector <string> forTypesIds = getForTypeIds();
  for (uint i=0;i<forTypesIds.size();i++){
    string ft = forTypesIds[i];
    for(uint u=0;u<diamClasses.size();u++){
      string dc = diamClasses[u];
      bool possible = false;
      for (int p=0; p<primProd_h.size();p++){
          string primProd = primProd_h[p];
          if(assessProdPossibility(primProd,ft, dc)){
              possible = true;
          }
      }
      if(possible){
        for (uint p=0;p<regPx.size();p++){
          Pixel* px = regPx[p];
          toReturn += px->vol_l.at(i).at(u)*px->avalCoef;
        }
      }
    }
  }
  return toReturn;
}

// ========================== LLData =======================================

LLData::LLData(ThreadManager* MTHREAD_h, string tableName_h){
  MTHREAD = MTHREAD_h;
  tableName = tableName_h;
}

LLData::~LLData(){

}

void
LLData::clean(){

  //checking the size is correct...
  int hsize = headers.size();
  for (uint i=0;i<records.size();i++){
    if(records[i].size() != hsize){
      vector <string> record = records[i];
      msgOut(MSG_CRITICAL_ERROR,"Error in the input reading table "+tableName+". Record "+i2s(i)+" has "+i2s(records[i].size())+" fields instead of "+i2s(hsize)+".");
    }
  }
  //cleaning empty-header columns...
  for (int i=headers.size()-1;i>=0;i--){
    if(headers[i] == ""){
      headers.erase(headers.begin()+i);
      for (uint j=0;j<records.size();j++){
        records[j].erase(records[j].begin()+i);
      }
    }
  }

}

string
LLData::getData(const int &pos_h, const string &header_h, const int &debugLevel) const{

  if (records.size()<= pos_h){
    msgOut(debugLevel, "Requested position "+i2s(pos_h)+" too high! Not enought records !!");
    return "";
  }
  int hsize = headers.size();
  for (uint i=0;i<hsize;i++){
    if(headers[i] == header_h) return records[pos_h][i];
  }
  msgOut(debugLevel, "Header string "+header_h+" not found!");
  return "";
}
```

5.99 /home/lobianco/git/ffsm_pp/src/ModelData.h File Reference

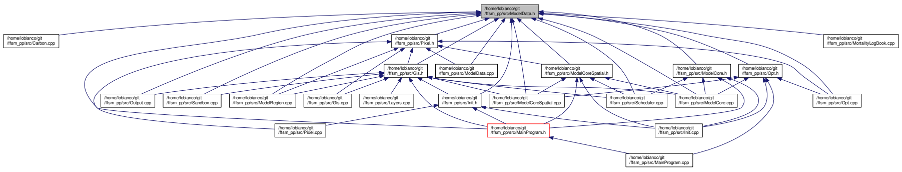
```
#include <string>
```

```
#include <vector>
#include <map>
#include <stdexcept>
#include <iostream>
#include <sstream>
#include <QString>
#include "BaseClass.h"
#include "InputNode.h"
#include "Output.h"
```

Include dependency graph for ModelData.h:



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



Classes

- struct scenarioData
- class ModelData

    Regional data, including macros and settings.
- struct IFiles

    Input files (struct)
- struct BasicData

    Basic data units (struct)
- struct forToProd

    IO production matrix between the forest resources and the primary products (struct)
- struct forType

    Forest types (struct)
- struct reclRule

    IO production matrix between the forest resources and the primary products (struct)
- struct pathRule

    Pathogen rule (how pathogen presense influence mortality) for a given forest type and diameter class (struct)
- class LLData

    Low level data. XML input is reversed here after unzipping oocalc file and parsing content.xml.

Typedefs

- typedef map< string, vector< double > > DataMap
- typedef pair< string, vector< double > > DataPair

5.99.1 Typedef Documentation

5.99.1.1 typedef map<string, vector <double> > DataMap

Definition at line 43 of file ModelData.h.

5.99.1.2 typedef pair<string, vector <double> > DataPair

Definition at line 44 of file ModelData.h.

5.100 ModelData.h

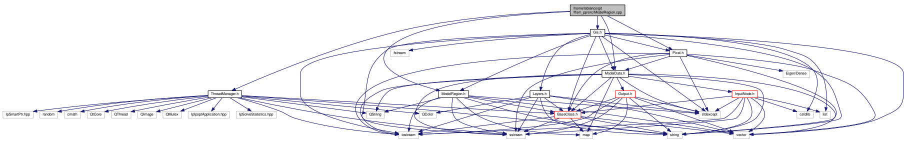
```
00001 /***************************************************************************
00002  *   Copyright (C) 2015 by Laboratoire d’Economie Forestière               *
00003  *   http://ffsm-project.org                                               *
00004  *                                                                         *
00005  *   This program is free software; you can redistribute it and/or modify  *
00006  *   it under the terms of the GNU General Public License as published by  *
00007  *   the Free Software Foundation; either version 3 of the License, or     *
00008  *   (at your option) any later version, given the compliance with the     *
00009  *   exceptions listed in the file COPYING that is distribued together     *
00010  *   with this file.                                                       *
00011  *                                                                         *
00012  *   This program is distributed in the hope that it will be useful,       *
00013  *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
00014  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
00015  *   GNU General Public License for more details.                          *
00016  *                                                                         *
00017  *   You should have received a copy of the GNU General Public License     *
00018  *   along with this program; if not, write to the                         *
00019  *   Free Software Foundation, Inc.,                                       *
00020  *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
00021  ***************************************************************************/
00022 #ifndef MODELDATA_H
00023 #define MODELDATA_H
00024 
00025 // Core C++ headers
00026 #include <string>
00027 #include <vector>
00028 #include <map>
00029 #include <stdexcept>
00030 #include <iostream>
00031 #include <sstream>
00032 
00033 // Qt headers...
00034 #include <QString>
00035 
00036 // RegMAS headers...
00037 #include "BaseClass.h"
00038 #include "InputNode.h"
00039 #include "Output.h"
00040 
00041 using namespace std;
00042 
00043 typedef map<string, vector <double> > DataMap;
00044 typedef pair<string, vector <double> > DataPair;
00045 
00046 struct IFiles;
00047 struct BasicData;
00048 class  LLData;
00049 class  ModelRegion;
00050 class  Layers;
00051 class  Output;
00052 struct forData;
00053 struct prodData;
00054 struct forToProd;
00055 struct reclRule;
00056 struct pathRule;
00057 struct forType;
00058 
00059 
00060 struct scenarioData {
00061   string                                            id;
00062   string                                     shortDesc;
00063   string                                      longDesc;
00064   string                                  settingTable;
```

```
  string                                  forDataTable;
  string                                 prodDataTable;
  string                                forToProdTable;
  string                                     pathTable;
};

/// Regional data, including macros and settings

/**
All regional data are within this class. It may have linked other data-classes.
<br>On some variables ModelData has just the definition of the objects, but the values may change at the agent-level. This is why each agent has a "personal copy" of them.
@author Antonello Lobianco
*/

class ModelData: public BaseClass{

public:
            ModelData(ThreadManager* MTHREAD_h);
             ~ModelData();

  /// Unzip the OpenOffice input file (NEW 2008.05.13)
  void                  loadInput();

  void                  loadDataFromCache(string tablename); ///< Load data from a cached CSV instead of the openoffice file
  vector<string>        getScenarios();
  int                   getScenarioIndex();
  bool                  delDir(QString dirname); ///< Recursivelly delete a directory
  void                  setScenarioData(); ///< Set the infos about this scenario (long description and overriding tables)
  void                  setDefaultSettings();
  void                  setScenarioSettings();
  void                  createRegions();
  void                  setDefaultForData();
  void                  setScenarioForData();
  void                  setDefaultProdData();
  void                  setScenarioProdData();
  void                  setDefaultProductResourceMatrixLink();
  void                  setScenarioProductResourceMatrixLink();
  void                  setForestTypes();
  void                  setReclassificationRules();
  void                  setDefaultPathogenRules();
  void                  setScenarioPathogenRules();
  void                  applyOverrides(); ///< Cancel all reg1 level data and trasform them in reg2 level if not already existing
  void                  applyDebugMode(); ///< Works only a specified subset of regions and products
  void                  setSpace();

  /// Return a vector of objects that together provide the specified resource in the specified quantity
  string                getOutputDirectory() const {return outputDirname;}
  int                   getFilenamesByDir (const string & dir, vector<string> &files, const string &filter = ""); ///< Return a list of files in a directory
  string                getFilenameByType(string type_h);
  LLData                getTable(string tableName_h, int debugLevel=MSG_CRITICAL_ERROR);
  vector <IFiles>       getIFilesVector() const {return iFilesVector;}
  string                getBaseDirectory() const {return baseDirectory;}
  ModelRegion*          getRegion(int regId_h);
  bool                  regionExist (const int & regId_h) const ;
  vector <ModelRegion*> getAllRegions(bool excludeResidual=true);
  vector<int>           getRegionIds(int level_h, bool excludeResidual=true);
  vector < vector <int> > getRegionIds( bool excludeResidual=true);
  string                regId2RegSName (const int & regId_h) const ;
  int                   regSName2RegId (const string & regSName_h) const ;
  int                   getNForTypes(){return forTypes.size();}
  int                   getNReclRules(){return reclRules.size();}
  forType*              getForType(int position){return &forTypes[position];}
  forType*              getForType(string& forTypeId_h);
  int                   getForTypeCounter(string& forTypeId_h, bool all=false); ///< By default it doesn't return forTypes used only as input
  vector <string>       getForTypeIds(bool all=false); ///< By default it doesn't return forTypes used only as input
  string                getForTypeParentId(const string& forTypeId_h);
  vector<string>        getForTypeChilds(const string &forTypeId_h);
  vector<int>           getForTypeChilds_pos(const string &forTypeId_h, bool all=false);
  vector<string>        getForTypeParents();
  int                   getNForTypesChilds(const string& forTypeId_h);
  reclRule*             getReclRule(int position){return &reclRules[position];}
  vector <string>       getDiameterClasses(bool productionOnly=false);
  /// A simple function to assess if a specified product can be made by a certain forest type and diameter class
  const bool            assessProdPossibility(const string &prod_h, const string &forType_h, const string &dClass_h);
  const int             getMaxYearUsableDeathTimber(const string &prod_h, const string &forType_h, const string &dClass_h);
  const int             getMaxYearUsableDeathTimber();
  int                   setErrorLevel(int errorLevel_h){errorLevel=errorLevel_h;}
  bool                  getTempBool(){return tempBool;}
  vector < vector <int> > createCombinationsVector(const int& nItems); ///< Return a vector containing any possible combination of nItems items (including any possible subset). The returned vector has in each slot the items present in that specific combination.

  double                getTimedData(const vector <double> &dated_vector, const int& year_h) const; ///< Return the value for the specified year in a timelly ordered vector, taking the last value if this is smaller than the required position.
  void                  setTimedData(const double& value_h, vector<double> &dated_vector, const int& year_h, const int& MSG_LEVEL=MSG_WARNING);

  int                   getIntSetting          (const string &name_h, int position=0) const;
  double                getDoubleSetting       (const string &name_h, int position=0) const;
  string                getStringSetting       (const string &name_h, int position=0) const;
  bool                  getBoolSetting         (const string &name_h, int position=0) const;
  vector <int>          getIntVectorSetting    (const string &name_h) const;
  vector <double>       getDoubleVectorSetting (const string &name_h) const;
  vector <string>       getStringVectorSetting (const string &name_h) const;
  vector <bool>         getBoolVectorSetting   (const string &name_h) const;


  const double          getProdData(const string &type_h, const int& regId_h, const string &prodId_h, const int &year=DATA_NOW, const string &freeDim_h="");
  const double          getForData(const string &type_h, const int& regId_h, const string &forType_h, const string &freeDim_h, const int& year=DATA_NOW);


  void                  setProdData(const double& value_h, const string &type_h, const int& regId_h, const string &prodId_h, const int& year=DATA_NOW, const bool& allowCreate=false, const string &freeDim_h=""); // Remember default arguments must be at the end
  void                  setForData(const double& value_h, const string &type_h, const int& regId_h, const string &forType_h, const string &freeDim_h, const int& year=DATA_NOW, const bool& allowCreate=false); // Remember default arguments must be at the end

  string                makeKeyProdData(const string& parName, const string& regId, const string& prod, const string& freeDim="") const {return parName+"#"+regId+"#"+prod+"#"+freeDim+"#";}
  string                makeKeyForData(const string& parName, const string& regId, const string & forType, const string& diamClass) const {return parName+"#"+regId+"#"+forType+"#"+diamClass+"#";}
  void                  unpackKeyProdData(const string& key, string& parName, int &regId, string& prod, string& freeDim) const;
  void                  unpackKeyForData(const string& key, string& parName, int &regId, string& forType, string& diamClass) const;

  vector<pathRule*>     getPathMortalityRule(const string& forType, const string& dC); ///< Return the pathogen mortality rule(s) associated with a given ft and dc (plural as more than a single pathogen could be found)

  double                calculateAnnualisedEquivalent(double amount_h, int years_h); ///< Calculate the annual equivalent flow
  double                calculateAnnualisedEquivalent(double amount_h, double years_h); ///< Transform the double to the highest integer and call calculateAnnualisedEquivalent(double amount_h, int years_h)

  void                  setOutputDirectory(const char* output_dirname_h);
  void                  setBaseDiretory(string baseDirectory_h){baseDirectory=baseDirectory_h;}
  void                  addSetting(string name_h, vector <string> values_h, int type_h, string comment_h);
  void                  addSetting(string name_h, string value_h, int type_h, string comment_h);
  void                  cacheSettings(); ///< Called after input reading, it fix frequently used data;
  int                   getCachedInitialYear(){return cached_initialYear;}

  void                  setBasicData(const string &name_h, int value, int position=0);
  void                  setBasicData(const string &name_h, double value, int position=0);
  void                  setBasicData(const string &name_h, string value, int position=0);
  void                  setBasicData(const string &name_h, bool value, int position=0);
  friend void           Output::printForestData(bool finalFlush=false);
  friend void           Output::printProductData(bool finalFlush=false);
  void                  deathTimberInventory_incrOrAdd(const iisskey &thekey, double value_h){incrOrAddMapValue(deathTimberInventory,thekey, value_h);}
  void                  deathTimberInventory_incr(const iisskey &thekey, double value_h){incrMapValue(deathTimberInventory,thekey, value_h);}
  double                deathTimberInventory_get(const iisskey &thekey){return findMap(deathTimberInventory, thekey);}
  map<iisskey, double > *  getDeathTimberInventory() {return &deathTimberInventory;};
  double                getAvailableDeathTimber(const vector<string> &primProd_h, int regID_h, int year_h); ///< Returns the timber available for a given set of primary products as stored in the deathTimberInventory map
  double                getAvailableAliveTimber(const vector<string> &primProd_h, int regId_h); ///< Returns the timber available for a given set of primary products as stored in the px->vol_l vector
  vector <int>          getAllocableProductIdsFromDeathTimber(const int &regId_h, const string &ft, const string &dc, const int &harvesting_year, int request_year=DATA_NOW); ///< Returns the ids of the primary products that is possible to obtain using the timber recorded death in the specific year, ft, dc combination
  scenarioData          scenario;

private:
  string              getBaseData (const string &name_h, int type_h, int position=0);
  vector <string>     getVectorBaseData (const string &name_h, int type_h);
  void                setBasicData(const string &name_h, string value, int type_h, int position);

  bool                dataMapCheckExist(const DataMap& map, const string& search_for, const bool& exactMatch=true) const;
  double              dataMapGetValue(const DataMap& map, const string& search_for, const int& year_h, const bool& exactMatch=true);
  int                 dataMapSetValue(DataMap& map, const string& search_for, const double& value_h, const int& year_h, const bool& exactMatch=true);

  string                                  inputFilename; // from Qt fileOpen dialog
  string                                  outputDirname; // from main config files
  string                                  baseDirectory; // from Qt fileOpen dialog

  map <string, vector<double> >              forDataMap; ///< Forestry data
  map <string, vector<double> >             prodDataMap; ///< Product data
  vector <forToProd>                    forToProdVector; ///< Vector of coefficients from forest resources to primary products

  vector <IFiles>                           iFilesVector; ///< List of all input files. Simple (struct)
  vector <BasicData>               programSettingsVector; ///< Setting data. Simple (struct)
  vector <LLData>                           LLDataVector; ///< Vector of Low Level Data
  vector <ModelRegion>                    regionsVector; ///< Vector of modelled regions

  vector <forType>                            forTypes; ///< Vector of forest types
  vector <reclRule>                          reclRules; ///< Vector of reclassification rules
  vector <pathRule>                          pathRules; ///< Vector of pathogen rules
  vector < vector <int> >                          l2r; ///< Region2 ids
  map<iisskey, double >            deathTimberInventory; ///< Map that register the death of biomass still usable as timber by year, l2_region, forest type and diameter class [Mm^3 wood]

  // cahced setting data..
  vector <string>                          diamClasses; ///< Diameter classes
  int                               cached_initialYear;
  vector <string>                          priProducts;
  vector <string>                          secProducts;
  vector <string>                          allProducts;

  bool                                       tempBool; ///< a temporary bool variable used for various functions

   /// For each agricultural soil type (as defined in the setting "agrLandTypes") this list define the objects that can be placed on that soil type
  InputNode                             mainDocument;  ///< the main input document, loaded in memory at unzipping stage
  int                                      errorLevel;
};

/// Input files (struct)
/**
Very short struct containing the input files used (one istance==one file).
<br>A copy of each Istances is saved on vector iFilesVector in class ModelData.
<br>iFiles are defined in the main config file and parsed subsequently.
@author Antonello Lobianco
*/
//Changed from a class to a structure on 2006.10.17.
struct IFiles {
  string                                   directory;
  string                                        type;
  string                                        name;
  string                                     comment;
};

/// Basic data units (struct)
/**
Struct containing the basic data objects. At the moment, data are used to store programm settings or macro data.
@author Antonello Lobianco
*/
struct BasicData {
  string                                        name;
  /// Values are stored as "string" because we don't yet know at this point if they are string, double or integers!
  vector <string>                             values;
  int                                           type; //< enum TYPE_*
  string                                     comment;
};

/// IO production matrix between the forest resources and the primary products (struct)
/**
Struct containing the io matrix between the forest resources and the primary products. Not to be confunded with the IO matrix between primary products and secondary products.
*/
struct forToProd {
  string                                        product;
  string                                        forType;
  string                                         dClass;
 /// The maximum year for a tree collapse that this product can be harvested from. E.g. a 0 value means it can be obtained only from live trees, a 5 years value mean it can be obtained from trees death no longer than 5 years ago.
  int                                          maxYears;
};

/// Forest types (struct)
/**
Struct containing the list of the forest types managed in the model.
@par memType Parameter to define if this type is used only in initial data reading, then is reclassed and no more used (1) or if it is generated from the reclass rule and then used in the model (2). New 20150311: (3) means a layer with spatial data of vol and area added respectiely in layers and proportionally to volumes
*/
struct forType {
  string                                        forTypeId;
  string                                         forLabel;
  int                                             memType;
  string                                         forLayer;
  string                                   ereditatedFrom;
  Layers*                                           layer;
};

/// IO production matrix between the forest resources and the primary products (struct)
/**
Struct containing the io matrix between the forest resources and the primary products. Not to be confunded with the IO matrix between primary products and secondary products.
*/
struct reclRule {
  int                                             regId;
  string                                      forTypeIn;
  string                                     forTypeOut;
  double                                          coeff;
};

/// Pathogen rule (how pathogen presense influence mortality) for a given forest type and diameter class (struct)
/**
Struct containing the rule that affect the mortality of a given ft and dc by a given pathogen: depending on the number of year of presence
of the pathogen over a given tollerance level the mortality increase more and more.
*/
struct pathRule {
  string                                      forType;
  string                                       dClass;
  string                                       pathId;  ///< Pathogen id (name)
  double                                      pres_min;  ///< Minimum level of presence of the pathogen to be counted as present (tolerance threshold)
  vector<double>                       mortCoefficents;  ///< Mortality coefficients ordered by number of presence of the pathogen, e.g. first value is the mortality increase in the first year of pathogen appareance.
};



/// Low level data. XML input is reversed here after unzipping oocalc file and parsing content.xml
class LLData: public BaseClass{

public:
                      LLData(ThreadManager* MTHREAD_h, string tableName_h);
                     ~LLData();
  void                clean(); // clean the data from empty headers
  string              getTableName(){return tableName;}
  int                 nrecords(){return records.size();}
  int                 nheaders(){return headers.size();}
  string              getData(const int& pos_h, const string& header_h, const int& debugLevel= MSG_CRITICAL_ERROR) const;
  friend void ModelData::loadInput();
  friend void ModelData::loadDataFromCache(string tablename);

private:
  string                             tableName;
  vector<string>                       headers;
  vector < vector <string> >           records;

};


#endif
```

5.101 /home/lobianco/git/ffsm_pp/src/ModelRegion.cpp File Reference

```
#include "ThreadManager.h"
#include "ModelRegion.h"
#include "ModelData.h"
#include "Pixel.h"
#include "Gis.h"
```

Include dependency graph for ModelRegion.cpp:



5.102 ModelRegion.cpp

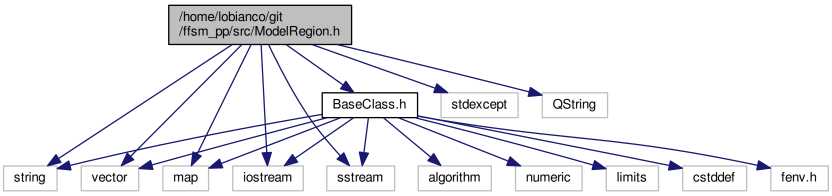
```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d'Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/

#include "ThreadManager.h"
#include "ModelRegion.h"
#include "ModelData.h"
#include "Pixel.h"
#include "Gis.h"


/**
The constructor of REGION instances want:
@param MTHREAD_h Pointer to the main thread manager
*/
ModelRegion::ModelRegion(ThreadManager* MTHREAD_h, int regId_h, string regSName_h, string regLName_h, int regLevel_h, int parRegId_h, bool isResidual_h){
  MTHREAD=MTHREAD_h;
  regId = regId_h;
  regSName = regSName_h;
  regLName = regLName_h;
  regLevel = regLevel_h;
  parRegId = parRegId_h;
  isResidual =  isResidual_h;

  // Create an empty vector of inventory bounds for each possible primary products combination
  int nInBounds = pow(2,MTHREAD->MD->getStringVectorSetting("priProducts").size());
  //int nInBounds = MTHREAD->MD->getStringVectorSetting("priProducts").size(); // TODO todo !Important
  vector <double> inBounds(nInBounds,0.); // should have ceated a vector of size priProducts.size(), all filled with zeros
  inResByAnyCombination             = inBounds;
  inResByAnyCombination_deathTimber = inBounds;
}

ModelRegion::~ModelRegion(){
}

vector<ModelRegion*>
```

```
ModelRegion::getChildren(bool excludeResidual){
  if(excludeResidual){
    vector<ModelRegion*> toReturn;
    for(uint i=0;i<chRegions.size();i++){
      if(!chRegions[i]->getIsResidual()){
        toReturn.push_back(chRegions[i]);
      }
    }
    return toReturn;
  }
  return chRegions;
}

int
ModelRegion::getNChildren(bool excludeResidual){
  if(excludeResidual){
    int toReturn;
    for(uint i=0;i<chRegions.size();i++){
      if(!chRegions[i]->getIsResidual()){
        toReturn++;
      }
    }
    return toReturn;
  }
  return chRegions.size();
}



double
ModelRegion::getVolumes(){
  /// \todo Implement me (but really needed?)
  return 0;
}

vector<double>
ModelRegion::getVolumes(int fType_h){
  /// \todo Implement me (but really needed?)
  vector<double> toReturn;
  return toReturn;
}

vector < vector <double> >
ModelRegion::getVolumes(int fType_h, string dClass_h){
  /// \todo Implement me (but really needed?)
  vector < vector <double> > toReturn;
  return toReturn;
}


double
ModelRegion::getArea(const string &fType_h, const string &dClass_h){
  vector <string> dClasses = MTHREAD->MD->getStringVectorSetting("dClasses");
  vector <string> fTypes= MTHREAD->MD->getForTypeIds();
  int ft_pos = -1000;
  int dc_pos = -1000;
  for(uint j=0;j<fTypes.size();j++){
    if (fTypes[j] == fType_h){
      ft_pos = j;
      break;
    }
  }
  for(uint u=0;u<dClasses.size();u++){
    if (dClasses[u] == dClass_h){
      dc_pos = u;
      break;
    }
  }
  if(ft_pos<0) msgOut(MSG_CRITICAL_ERROR,"Forest type "+fType_h+" not found in getArea() function.");
  if(dc_pos<0) msgOut(MSG_CRITICAL_ERROR,"Diameter class"+dClass_h+" not found in getArea() function.");

  return getArea(ft_pos, dc_pos);
}

double
ModelRegion::getArea(const string &fType_h){
  vector <string> fTypes= MTHREAD->MD->getForTypeIds();
  int ft_pos = -1000;
  for(uint j=0;j<fTypes.size();j++){
    if (fTypes[j] == fType_h){
      ft_pos = j;
      break;
    }
  }
  if(ft_pos<0) msgOut(MSG_CRITICAL_ERROR,"Forest type "+fType_h+" not found in getArea() function.");
  return getArea(ft_pos);
}

double
ModelRegion::getArea(const int& ft_pos, const int& dc_pos){
  double totalarea = 0.0;
  for(uint i=0;i<myPixels.size(); i++){
     totalarea += myPixels[i]->area.at(ft_pos).at(dc_pos);
  }
  return totalarea;
}

double
ModelRegion::getArea(const int& ft_pos){
  double totalarea = 0.0;
  for(uint i=0;i<myPixels.size(); i++){
     totalarea += vSum(myPixels[i]->area.at(ft_pos));
  }
  return totalarea;
}

double
ModelRegion::getArea(){
  vector<Pixel*> regPx = this->getMyPixels();
  double totalarea = 0.0;
  for(uint i=0;i<myPixels.size(); i++){
     totalarea += vSum(myPixels[i]->area);
  }
  return totalarea;
}

double
ModelRegion::getValue(string layerName, int op){
  int nPx = myPixels.size();
  double sumvalue=0;
  for(uint i=0;i<nPx; i++){
    sumvalue += myPixels[i]->getDoubleValue(layerName,true);
  }
  if(op==OP_SUM){
    return sumvalue;
  } else if (op == OP_AVG) {
    return sumvalue/nPx;
  } else {
    string thisf = __PRETTY_FUNCTION__;
    msgOut(MSG_CRITICAL_ERROR, "in "+thisf+", operation not supported");
  }
  return 0.;
}

/***
 * It establishs a link between pixels and regions.
 *
 * Pixels are stored in myPixels vectors and, only if this is a level2 region, a pointer to this region is passed to the pixel
 *
 * */
void
ModelRegion::setMyPixels(){
  int xyNPixels = MTHREAD->GIS->getXyNPixels();
  for(uint i=0;i<xyNPixels;i++){
    Pixel* px = MTHREAD->GIS->getPixel(i);
    if(px->getDoubleValue("regLev_1")==regId || px->getDoubleValue("regLev_2")==regId){
      myPixels.push_back(px);
      if(regLevel == 2){
        px->setMyRegion(this);
      }
    }
  }
}

void
ModelRegion::swap(const int& swap_what){

  for(uint i=0;i<myPixels.size();i++) {
    myPixels[i]->swap(swap_what);
  }

}



```

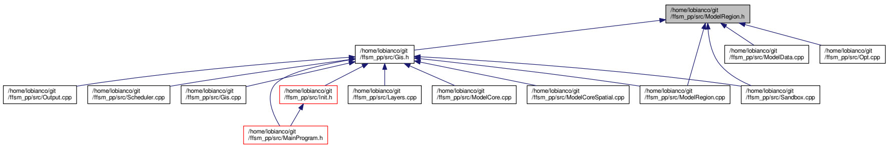
5.103 /home/lobianco/git/ffsm_pp/src/ModelRegion.h File Reference

```
#include <string>
#include <vector>
#include <map>
#include <stdexcept>
#include <iostream>
#include <sstream>
#include <QString>
#include "BaseClass.h"
```

Include dependency graph for ModelRegion.h:



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



Classes

- class ModelRegion

5.104 ModelRegion.h

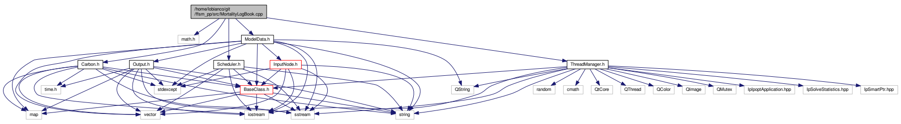
```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d'Economie Forestière              *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#ifndef MODELREGION_H
```

```
#define MODELREGION_H

// Core C++ headers
#include <string>
#include <vector>
#include <map>
#include <stdexcept>
#include <iostream>
#include <sstream>

// Qt headers...
#include <QString>

// regmas headers..
#include "BaseClass.h"

using namespace std;

struct forData;
struct prodData;
class Pixel;

class ModelRegion : public BaseClass{

public:
            ModelRegion(ThreadManager* MTHREAD_h, int regId_h, string regSName_h,
string regLName_h, int regLevel_h, int parRegId_h, bool isResidual_h); ///< Constructor
             ~ModelRegion();

  // "set" methods..
  void                setRegId(int regId_h){regId = regId_h;};
  void                setRegSName(string regSName_h){regSName = regSName_h;};
  void                setRegLName(string regLName_h){regLName = regLName_h;};
  void                setRegLevel(int regLevel_h){regLevel = regLevel_h;};
  void                setParRegId(int parRegId_h){parRegId = parRegId_h;};
  void                setIsResidual(bool isResidual_h){isResidual = isResidual_h;};
  void                setParent(ModelRegion* parRegion_h){parRegion = parRegion_h;};
  void                setChildren(vector<ModelRegion*> children_h) {chRegions = children_h;};
///< Childrens are all the lvel-1 region that are parts of this region.
  void                addForData(forData* data_h){forDataVector.push_back(data_h);};
  void                addProdData(prodData* data_h){prodDataVector.push_back(data_h);};
  void                setMyPixels(); ///< It sets a double link pixels <--> region
  void                swap(const int& swap_what);

  // "get" methods..
  int                 getRegId()      const {return regId;};
  string              getRegSName()   const {return regSName;};
  string              getRegLName()   const {return regLName;};
  int                 getRegLevel()   const {return regLevel;};
  int                 getParRegId()   const {return parRegId;};
  bool                getIsResidual() const {return isResidual;};
  ModelRegion*        getParent(){return parRegion;}; ///< Returns a pointer to the
 parent regions
  vector<ModelRegion*>getChildren(bool excludeResidual = true); ///< Return a vector of pointers to the
 direct child regions
  double              getVolumes();
  vector<double>      getVolumes(int fType_h);
  double              getValue(string layerName, int op=OP_SUM); ///< return the values of its own
 pixels for the specified layer. Possible operations: OP_SUM or OP_AVG
  vector < vector <double> > getVolumes(int fType_h, string dClass_h);
  double              getArea(const string &fType_h, const string &dClass_h); ///< Get area by ft and dc
 (from pixel->area matrix)
  double              getArea(const string &fType_h); ///< Get area by ft (from pixel->area matrix)
  double              getArea(const int& ft_pos, const int& dc_pos); ///< Get area by ft and dc positions
 (from pixel->area matrix)
  double              getArea(const int& ft_pos); ///< Get area by ft position (from pixel->area matrix)
  double              getArea(); ///< Get whole forest area (from pixel->area matrix)

  int                 getNChildren(bool excludeResidual = true);
  vector<Pixel*>      getMyPixels(){return myPixels;};

  vector<double>     inResByAnyCombination;             ///< Vector of inventory
 resource for each possible combination of primary products. This store both alive timber and death one.
  vector<double>     inResByAnyCombination_deathTimber; ///< Vector of
 inventory resource for each possible combination of primary products. This store only death timber.

private:
  int                                  regId; ///< Regional unique ID
  string                            regSName; ///< A short name of the region
  string                            regLName; ///< Region long name;
  int                               regLevel; ///< The level of the region. 1: country, 2: regions
  int                               parRegId; ///< Id of the parent region;
  bool                            isResidual; ///< A flag if this region should be explicitelly
 modelled or it is just a residual
  ModelRegion*                     parRegion; ///< Pointer to the parent region
  vector<ModelRegion*>             chRegions; ///< Vector of level-1 children regions
  vector<forData*>             forDataVector; ///< Vector of pointers of forestry data (owned by
    ModelData)
   vector<prodData*>         prodDataVector; ///< Vector of pointers of product data (owned by
    ModelData)
   vector<Pixel*>                  myPixels; ///< Vector of pixels for this region



};

#endif // REGION_H
```

5.105 /home/lobianco/git/ffsm_pp/src/MortalityLogBook.cpp File Reference

```
#include <math.h>
#include "Carbon.h"
#include "ThreadManager.h"
#include "ModelData.h"
#include "Scheduler.h"
```

Include dependency graph for MortalityLogBook.cpp:



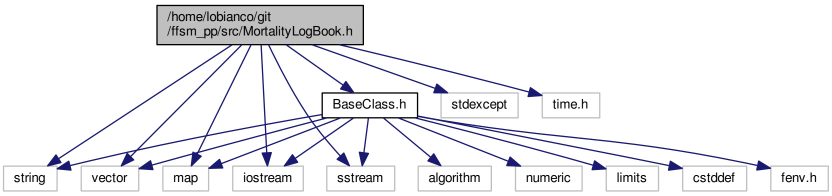
5.106 MortalityLogBook.cpp

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d'Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify *
 *   it under the terms of the GNU General Public License as published by *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/

#include <math.h>       /* log */

#include "Carbon.h"
#include "ThreadManager.h"
#include "ModelData.h"
#include "Scheduler.h"



Carbon::Carbon(ThreadManager* MTHREAD_h){
  MTHREAD=MTHREAD_h;
}

Carbon::~Carbon(){
}


// ################# GET FUNCTIONS ################
/**
 * @param reg
 * @param stock_type
 * @return the Carbon stocked in a given sink
 *
 * For product sink:
 * - for primary products it includes the primary products exported out of the country, but not those exported to other regions or used in the region as
 *   these are assumed to be totally transformed to secondary products;
 * - for secondary products it includes those produced in the region from locally or regionally imported primary product plus those secondary products
 *   imported from other regions, less those exported to other regions. It doesn't include the secondary products imported from abroad the country.
 */
double
Carbon::getStock(const int & regId, const int & stock_type) const{
  double toReturn = 0.0;
  int currentYear = MTHREAD->SCD->getYear();
  int initialYear = MTHREAD->MD->getIntSetting("initialYear");
  switch (stock_type){
    case STOCK_PRODUCTS: {
      vector <string> priProducts = MTHREAD->MD->getStringVectorSetting("priProducts");
      vector <string> secProducts = MTHREAD->MD->getStringVectorSetting("secProducts");
      vector <string> allProducts = priProducts;
      allProducts.insert( allProducts.end(), secProducts.begin(), secProducts.end() );
      for(uint i=0;i<allProducts.size();i++){
        double coeff = MTHREAD->MD->getProdData("co2content_products",regId,allProducts[i],DATA_NOW,""); //  [kg CO2/m^3 wood]
        double life  = MTHREAD->MD->getProdData("avgLife_products",regId,allProducts[i],DATA_NOW,"");
        //for(int y=currentYear;y>currentYear-life;y--){ // ok
        //  iiskey key(y,regId,allProducts[i]);
        //  toReturn += findMap(products,key,MSG_NO_MSG,0.0)*coeff/1000;
        //}
        for(int y=(initialYear-100);y<=currentYear;y++){
          iiskey key(y,regId,allProducts[i]);
          double originalStock = findMap(products,key,MSG_NO_MSG,0.0);
          double remainingStock = getRemainingStock(originalStock,life,currentYear-y);
          toReturn += remainingStock*coeff/1000;
        }
      }
      break;
    }
    case STOCK_INV:{
      vector <string> fTypes = MTHREAD->MD->getForTypeIds();
      for(uint i=0;i<fTypes.size();i++){
        // units:
        // co2content_inventory: [Kg CO2 / m^3 wood]
        // co2content_extra:     [Kg CO2 / m^3 inventaried wood]
        double coeff = MTHREAD->MD->getForData("co2content_inventory",regId,fTypes[i],"",DATA_NOW); //  [kg CO2/m^3 wood]
        double life  = MTHREAD->MD->getForData("avgLive_deathBiomass_inventory",regId,fTypes[i],"",DATA_NOW);
        // PART A: from death biomass..
        //for(int y=currentYear;y>currentYear-life;y--){ // ok
        //  iiskey key(y,regId,fTypes[i]);
        //  toReturn += findMap(deathBiomassInventory,key,MSG_NO_MSG)*coeff/1000;
        //}
        for(int y=(initialYear-100);y<=currentYear;y++){
          iiskey key(y,regId,fTypes[i]);
          double originalStock = findMap(deathBiomassInventory,key,MSG_NO_MSG,0.0);
          double remainingStock = getRemainingStock(originalStock,life,currentYear-y);
          toReturn += remainingStock*coeff/1000;
        }

        // PART B: from inventory volumes
        toReturn += MTHREAD->MD->getForData("vol",regId,fTypes[i],DIAM_ALL,DATA_NOW)*coeff/1000;
      }
      break;

    }
    case STOCK_EXTRA:{
      vector <string> fTypes = MTHREAD->MD->getForTypeIds();
      for(uint i=0;i<fTypes.size();i++){
        // units:
        // co2content_inventory: [Kg CO2 / m^3 wood]
        // co2content_extra:     [Kg CO2 / m^3 inventaried wood]
        double coeff = MTHREAD->MD->getForData("co2content_extra",regId,fTypes[i],"",DATA_NOW); //  [kg CO2/m^3 wood]
        double life  = MTHREAD->MD->getForData("avgLive_deathBiomass_extra",regId,fTypes[i],"",DATA_NOW);
        // PART A: from death biomass..
        //for(int y=currentYear;y>currentYear-life;y--){ // ok
        //  iiskey key(y,regId,fTypes[i]);
        //  toReturn += findMap(deathBiomassExtra,key,MSG_NO_MSG),0.0*coeff/1000;
        //}
        for(int y=(initialYear-100);y<=currentYear;y++){
          iiskey key(y,regId,fTypes[i]);
          double originalStock = findMap(deathBiomassExtra,key,
MSG_NO_MSG,0.0);
          double remainingStock = getRemainingStock(originalStock,life,currentYear-y);
          toReturn += remainingStock*coeff/1000;
        }
        // PART B: from inventory volumes
        double extraBiomass_ratio = MTHREAD->MD->getForData("extraBiomass_ratio",regId,
fTypes[i],"",DATA_NOW);
        toReturn += MTHREAD->MD->getForData("vol",regId,fTypes[i],
DIAM_ALL,DATA_NOW)*extraBiomass_ratio*coeff/1000;
      }
      break;
    }
    default:
      msgOut(MSG_CRITICAL_ERROR,"Unexpected stock_type in function getStock");
  }
  return toReturn;
}


double
Carbon::getCumSavedEmissions(const int & regId, const int & em_type) const{
  switch (em_type){
    case EM_ENSUB:
      return findMap(cumSubstitutedEnergy, regId);
      break;
    case EM_MATSUB:
      return findMap(cumSubstitutedMaterial, regId);
      break;
    case EM_FOROP:
      return -findMap(cumEmittedForOper, regId);
      break;
    default:
      msgOut(MSG_CRITICAL_ERROR,"Unexpected em_type in function
 getCumSavedEmissions");
  }
  return 0.0;
}

// ################# INITIALISE FUNCTIONS ################

void
Carbon::initialiseEmissionCounters(){
  vector<int> regIds = MTHREAD->MD->getRegionIds(2);
  for (uint i=0;i<regIds.size();i++){
    pair<int,double> mypair(regIds[i],0.0);
    cumSubstitutedEnergy.insert(mypair);
    cumSubstitutedMaterial.insert(mypair);
    cumEmittedForOper.insert(mypair);
  }
}

void
Carbon::initialiseDeathBiomassStocks(const vector<double> & deathByFt,
const int & regId){
  // it must initialize in the past the death biomass taking the value of the first year
  vector <string> fTypes = MTHREAD->MD->getForTypeIds();
  if(fTypes.size() != deathByFt.size()) {msgOut(MSG_CRITICAL_ERROR,"deathByFt and
 fTypes have different lenght!");}
  int currentYear = MTHREAD->SCD->getYear();
  //int initialYear = MD->getIntSetting("initialYear");

  for(uint i=0;i<fTypes.size();i++){
//    double life_inventory     =
 MTHREAD->MD->getForData("avgLive_deathBiomass_inventory",regId,fTypes[i],"",DATA_NOW);
//    double life_extra         =
 MTHREAD->MD->getForData("avgLive_deathBiomass_extra",regId,fTypes[i],"",DATA_NOW);
      double extraBiomass_ratio = MTHREAD->MD->getForData("extraBiomass_ratio",regId,
fTypes[i],"",DATA_NOW);

//    for(int y=currentYear;y>currentYear-life_inventory;y--){
//        iiskey key(y,regId,fTypes[i]);
//        pair<iiskey,double> mypair(key,deathByFt.at(i));
//        deathBiomassInventory.insert(mypair);
//    }
//    for(int y=currentYear;y>currentYear-life_extra;y--){
//        iiskey key(y,regId,fTypes[i]);
//        pair<iiskey,double> mypair(key,deathByFt.at(i)*extraBiomass_ratio);
//        deathBiomassExtra.insert(mypair);
//    }

    for(int y=currentYear;y>currentYear-100;y--){
        iiskey key(y,regId,fTypes[i]);
        pair<iiskey,double> mypairInventory(key,deathByFt.at(i));
        pair<iiskey,double> mypairExtra(key,deathByFt.at(i)*extraBiomass_ratio);
        deathBiomassInventory.insert(mypairInventory);
        deathBiomassExtra.insert(mypairExtra);
    }
  }
}

void
Carbon::initialiseProductsStocks(const vector<double> & qByProduct, const int & regId){
  // it must initialize in the past the products taking the value of the first year
  vector <string> priProducts = MTHREAD->MD->getStringVectorSetting("priProducts");
  vector <string> secProducts = MTHREAD->MD->getStringVectorSetting("secProducts");
  vector <string> allProducts = priProducts;
  allProducts.insert( allProducts.end(), secProducts.begin(), secProducts.end() );
  if(allProducts.size() != qByProduct.size()) {msgOut(MSG_CRITICAL_ERROR,"allProducts and qByProduct have different lenght!");}
  int currentYear = MTHREAD->SCD->getYear();
  for(uint i=0;i<allProducts.size();i++){
    double life  = MTHREAD->MD->getProdData("avgLife_products",regId,allProducts[i],DATA_NOW);
    //for(int y=currentYear;y>currentYear-life;y--){
    for(int y=currentYear;y>currentYear-100;y--){
        iiskey key(y,regId,allProducts[i]);
        pair<iiskey,double> mypair(key,qByProduct.at(i));
        products.insert(mypair);
    }
  }
  //cout << "" << endl;
}

// ################# REGISTER FUNCTIONS ################
void
Carbon::registerHarvesting(const double & value, const int & regId, const string & fType){
  double convCoeff = MTHREAD->MD->getForData("forOperEmissions",regId,fType,""); //  Kg of CO2 emitted per cubic meter of forest operations
  // units:
  // value: Mm^3
  // convCoeff: Kg CO2/m^3 wood
  // desidered output: Mt CO2
  // ==> I must divide by 1000
  addSavedEmissions(-convCoeff*value/1000,regId,EM_FOROP);
  // Add the extraBiomass associated to the harvested volumes to the deathBiomassExtra pool
  int    year               = MTHREAD->SCD->getYear();
  double extraBiomass_ratio = MTHREAD->MD->getForData("extraBiomass_ratio",regId,fType,"",DATA_NOW);
  double newDeathBiomass = value*extraBiomass_ratio;
  iiskey key(year,regId,fType);
  incrOrAddMapValue(deathBiomassExtra, key, newDeathBiomass);
}


void
Carbon::registerDeathBiomass(const double &value, const int & regId, const string & fType){
  int year = MTHREAD->SCD->getYear();
  iiskey key(year,regId,fType);
  double extraBiomass_ratio = MTHREAD->MD->getForData("extraBiomass_ratio",regId,fType,"",DATA_NOW);
  //pair<iiskey,double> mypairInventory(key,value);
  //pair<iiskey,double> mypairExtra(key,value*extraBiomass_ratio);
  incrOrAddMapValue(deathBiomassInventory, key, value);
  incrOrAddMapValue(deathBiomassExtra, key, value*extraBiomass_ratio);
  //deathBiomassInventory.insert(mypairInventory);
  //deathBiomassExtra.insert(mypairExtra);

}

void
Carbon::registerProducts(const double &value, const int & regId, const string & productName){
  // Registering the CO2 stock embedded in the product...
  int year = MTHREAD->SCD->getYear();
  iiskey key(year,regId,productName);
  pair<iiskey,double> mypair(key,value);
  products.insert(mypair);
  // registering the substituted CO2 for energy and material..
  double subEnergyCoeff   = MTHREAD->MD->getProdData("co2sub_energy",regId,productName,DATA_NOW,"");
  double subMaterialCoeff = MTHREAD->MD->getProdData("co2sub_material",regId,productName,DATA_NOW,"");
  // units:
  // value: Mm^3
  // subEnergyCoeff and subMaterialCoeff:  [kgCO2/m^3 wood]
  // desidered output: Mt CO2
  // ==> I must divide by 1000
  //addSavedEmissions(subEnergyCoeff*value/1000,regId,EM_ENSUB);
  addSavedEmissions(subMaterialCoeff*value/1000,regId,EM_MATSUB);
}



void
Carbon::registerTransports(const double &distQ, const int & regId){
  // units:
  // distQ: km*Mm^3
  // transportEmissionsCoeff:  [Kg CO2 / (km*m^3) ]
  // desidered output: Mt CO2
  // ==> I must divide by 1000
  double transportEmissionsCoeff = MTHREAD->MD->getDoubleSetting("
transportEmissionsCoeff");
  addSavedEmissions(-transportEmissionsCoeff*distQ/1000,regId,
EM_FOROP);
}

void
Carbon::HWP_eol2energy(){

  int currentYear = MTHREAD->SCD->getYear();
  int initialYear = MTHREAD->MD->getIntSetting("initialYear");
  vector <string> priProducts = MTHREAD->MD->getStringVectorSetting("
priProducts");
  vector <string> secProducts = MTHREAD->MD->getStringVectorSetting("
secProducts");
  vector <string> allProducts = priProducts;
  allProducts.insert( allProducts.end(), secProducts.begin(), secProducts.end() );

  vector<int> regIds = MTHREAD->MD->getRegionIds(2);
  for (uint r=0;r<regIds.size();r++){
    double regId = regIds[r];
    for(uint i=0;i<allProducts.size();i++){
      string pr = allProducts[i];
      double life  = MTHREAD->MD->getProdData("avgLife_products",regId,pr,
DATA_NOW,"");
      double eol2e_share      = MTHREAD->MD->getProdData("eol2e_share",regId,pr,
DATA_NOW,"");
      double subEnergyCoeff   = MTHREAD->MD->getProdData("co2sub_energy",regId,pr,
DATA_NOW,"");
      if(eol2e_share > 0 && subEnergyCoeff>0){
        for(int y=(initialYear-100);y<currentYear;y++){ // notice the minor operator and not minor equal:
 energy substitution for products produced this year assigned to the following year, otherwise double counring
 in the process of making dicrete the exponential function
          iiskey key(y,regId,pr);
          double originalStock = findMap(products,key,MSG_NO_MSG,0.0);
          double remainingStockLastYear = getRemainingStock(originalStock,life,currentYear
-y-1);
          double remainingStockThisYear = getRemainingStock(originalStock,life,currentYear
-y);
          double eofThisYear = remainingStockLastYear-remainingStockThisYear;
          addSavedEmissions(subEnergyCoeff*eofThisYear*eol2e_share/1000,regId,
EM_ENSUB);
        }
      }
    }
  }

}


// ################# UTILITY (PRIVATE) FUNCTIONS ################

void
Carbon::addSavedEmissions(const double & value, const int & regId, const int &
em_type){
  switch (em_type){
    case EM_ENSUB:
      incrMapValue(cumSubstitutedEnergy, regId, value);
      break;
    case EM_MATSUB:
      incrMapValue(cumSubstitutedMaterial, regId, value);
      break;
    case EM_FOROP:
      incrMapValue(cumEmittedForOper, regId, -value);
      break;
    default:
      msgOut(MSG_CRITICAL_ERROR,"Unexpected em_type in function
 getCumSavedEmissions");
  }
}

double
Carbon::getRemainingStock(const double & initialValue, const double & halfLife,
const double & years) const{
  // // TODO: remove this test
  //if(years>0) return 0.0;
  //return initialValue;

  double k  = log(2)/halfLife;
  return initialValue*exp(-k*years);
}

```

5.107 /home/lobianco/git/ffsm_pp/src/MortalityLogBook.h File Reference

```
#include <string>
#include <vector>
#include <map>
#include <stdexcept>
#include <iostream>
#include <sstream>
#include <time.h>
#include "BaseClass.h"
```

Include dependency graph for MortalityLogBook.h:



Classes

- class Carbon

  Class responsable to keep the logbook of the Carbon Balance.

5.108 MortalityLogBook.h

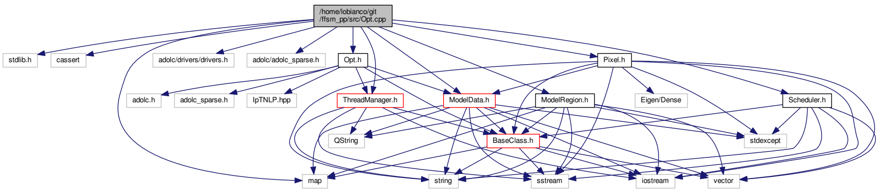
```
/***************************************************************************
 *   Copyright (C) 2016 by Laboratoire d'Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify *
 *   it under the terms of the GNU General Public License as published by *
 *   the Free Software Foundation; either version 3 of the License, or    *
 *   (at your option) any later version, given the compliance with the    *
 *   exceptions listed in the file COPYING that is distribued together    *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,      *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of       *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the        *
 *   GNU General Public License for more details.                         *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License    *
 *   along with this program; if not, write to the                        *
 *   Free Software Foundation, Inc.,                                      *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.            *
 ***************************************************************************/
#ifndef MORTALITYLOGBOOK_H
#define MORTALITYLOGBOOK_H
```

```

// Core C++ headers
#include <string>
#include <vector>
#include <map>
#include <stdexcept>
#include <iostream>
#include <sstream>
#include <time.h>

//regmas headers
#include "BaseClass.h"

/// Class responsable to keep the logbook of the Death Timber still usable by the market module
/**
@author Antonello Lobianco

A single istance of this class exists and is available trought the global MTHREAD->MLB pointer.

It consits of functions to track a mortality-related event and store the information in STL maps that register the events and keep updated the stocks.

Carbon pools are stored as Mm^3 wood while and emission cumulated counters are directly in Mt CO2.

getStock() and getCumSavedEmissions() are then used to report the current levels of carbon in the stock or emitted/substituted.
*/

class Carbon: public BaseClass{
public:
                        Carbon(ThreadManager* MTHREAD_h); ///< Constructor
                       ~Carbon();


  double                getStock(const int & regId, const int & stock_type) const;         ///< Returns the current stock of carbon [Mt CO2]
  double                getCumSavedEmissions(const int & regId, const int & em_type) const;                 ///< Returns the current cumulative saved emissions by type [Mt CO2]

  void                  registerHarvesting(const double & value, const int & regId, const string &fType);   ///< Registers the harvesting of trees increasing the value of cumEmittedForOper
  void                  registerDeathBiomass(const double &value, const int & regId, const string &fType);  ///< Registers the "death" of a given amount of biomass, storing it in the deathBiomass map
  void                  registerProducts(const double &value, const int & regId, const string &productName);///< Registers the production of a given amount of products, storing it in the products maps. Also increase material substitution.
  void                  registerTransports(const double &distQ, const int & regId);                   ///< Registers the quantities emitted by transport of wood FROM a given region
  void                  initialiseDeathBiomassStocks(const vector<double> & deathByFt, const int & regId);  ///< Initialises the stocks of death biomass for the avgLive_* years before the simulation starts
  void                  initialiseProductsStocks(const vector<double> & qByProduct, const int &regId);      ///< Initialises the stocks of products for the avgLive_* years before the simulation starts
  void                  initialiseEmissionCounters();                              ///< Initialises the emission counters to zero.
  void                  HWP_eol2energy();                 ///< Computes the energy substitution for the quota of HWP that reachs end of life and doesn't go to landfill


private:
  void                  addSavedEmissions(const double & value, const int & regId, const int & em_type);    ///< Increases the value to the saved emissions for a given type and region
  double                getRemainingStock(const double & initialValue, const double & halfLife, const double & years) const; ///< Apply a single exponential decay model to retrieve the remining stock given the initial stock, the half life and the time passed from stock formation.

  map<iiskey, double >     deathBiomassInventory; ///< Map that register the death of biomass by year, l2_region and forest type (inventoried)[Mm^3 wood]
  map<iiskey, double >         deathBiomassExtra; ///< Map that register the death of biomass by year, l2_region and forest type (non-inventoried biomass: branches, roots..) [Mm^3 wood]
  map<iiskey, double >                  products; ///< Map that register the production of a given product by year, l2_region and product [Mm^3 wood]
  map<int,double>           cumSubstitutedEnergy; ///< Map that store the cumulative CO2 substituted for energy consumption, by l2_region [Mt CO2]
  map<int,double>         cumSubstitutedMaterial; ///< Map that store the cumulative CO2 substituted using less energivory material, by l2_region [Mt CO2]
  map<int,double>              cumEmittedForOper; ///< Map that store emissions for forest operations, including transport, by l2_region [Mt CO2]



};

#endif // CARBON_H
```

5.109 /home/lobianco/git/ffsm_pp/src/Opt.cpp File Reference

```
#include <stdlib.h>
#include <cassert>
#include <map>
#include <adolc/drivers/drivers.h>
#include <adolc/adolc_sparse.h>
#include "Opt.h"
#include "ModelData.h"
#include "Pixel.h"
#include "ThreadManager.h"
#include "Scheduler.h"
#include "ModelRegion.h"
```

Include dependency graph for Opt.cpp:



Macros

- #define CONSTRAIN_START_LOOP(pVector, cn)
- #define CONSTRAIN_END_LOOP }}}

Typedefs

- typedef std::map< string, endvar > VarMap
- typedef std::pair< std::string, endvar > VarPair

5.109.1 Macro Definition Documentation

5.109.1.1 #define CONSTRAIN_END_LOOP }}}

Definition at line 46 of file Opt.cpp.

Referenced by Opt::eval_constraints().

5.109.1.2 #define CONSTRAIN_START_LOOP(*pVector,* *cn*)

Value:

```
for (uint r1=0;r1<l2r.size();r1++){ \
    for (uint r2=0;r2<l2r[r1].size();r2++){ \
      for (uint p=0;p<(pVector).size();p++){ \
        int psec = p+nPriPr; \
        cix = gix((cn), r1, r2, p);
```

Definition at line 40 of file Opt.cpp.

Referenced by Opt::eval_constraints().

5.109.2 Typedef Documentation

5.109.2.1 typedef std::map<string, endvar> VarMap

Definition at line 52 of file Opt.cpp.

5.109.2.2 typedef std::pair<std::string, endvar > VarPair

Definition at line 53 of file Opt.cpp.

5.110 Opt.cpp

```
00001 /***************************************************************************
00002  *   Copyright (C) 2015 by Laboratoire d'Economie Forestière               *
00003  *   http://ffsm-project.org                                               *
00004  *                                                                         *
00005  *   This program is free software; you can redistribute it and/or modify  *
00006  *   it under the terms of the GNU General Public License as published by  *
00007  *   the Free Software Foundation; either version 3 of the License, or     *
00008  *   (at your option) any later version, given the compliance with the     *
00009  *   exceptions listed in the file COPYING that is distribued together     *
00010  *   with this file.                                                       *
00011  *                                                                         *
00012  *   This program is distributed in the hope that it will be useful,       *
00013  *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
00014  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
00015  *   GNU General Public License for more details.                          *
00016  *                                                                         *
00017  *   You should have received a copy of the GNU General Public License     *
00018  *   along with this program; if not, write to the                         *
00019  *   Free Software Foundation, Inc.,                                       *
00020  *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
00021  ***************************************************************************/
00022 
00023 #include <stdlib.h>
00024 
00025 #include <cassert>
00026 
00027 #include <map>
00028 #include <adolc/drivers/drivers.h>
00029 #include <adolc/adolc_sparse.h>
00030 
00031 #include "Opt.h"
00032 
00033 #include "ModelData.h"
00034 #include "Pixel.h"
00035 #include "ThreadManager.h"
00036 #include "Scheduler.h"
00037 #include "ModelRegion.h"
00038 #include "Opt.h"
00039 
00040 #define CONSTRAIN_START_LOOP(pVector,cn) \
00041   for (uint r1=0;r1<l2r.size();r1++){ \
00042     for (uint r2=0;r2<l2r[r1].size();r2++){ \
00043       for (uint p=0;p<(pVector).size();p++){ \
00044         int psec = p+nPriPr; \
00045         cix = gix((cn), r1, r2, p);
00046 #define CONSTRAIN_END_LOOP \
00047   }}}
00048 
00049 
00050 using namespace Ipopt;
00051 
00052 typedef std::map<string, endvar>        VarMap;
00053 typedef std::pair<std::string, endvar > VarPair;
00054 
00055 // ****** MODEL IMPLEMENTATION START HERE ********************************
00056 
00057 
00058 void
00059 Opt::declareVariables(){
00060     // filling the list of variables and their domain and optionally their bonds
00061     // if you add variables in the model that enter optimisation you'll have to add them here
00062     // the underlying map goes automatically in alphabetical order
00063     // original order: pc,pl,dc,dl,da,sc,sl,sa,exp
00064     // 20140328: if these vars have a lower bound > 0 the model doesn't solve when volumes in a region go
```

```
 to zero !!!

    // syntax: declareVariable("name", domainType, lbound[default=0], ubound[default= +inf], variable defining lower bounds[default=""], variable defining upper bound[default=""])

    // all variables have upper or equal than zero bound:
    declareVariable("da", DOM_SEC_PR,    "Demand from abroad (imports)");
    declareVariable("dc", DOM_SEC_PR,    "Demand, composite");
    declareVariable("dl", DOM_ALL_PR,    "Demand from local");
    declareVariable("pc", DOM_ALL_PR,    "Price, composite");
    declareVariable("pl", DOM_ALL_PR,    "Price, local") ;
    declareVariable("rt", DOM_R2_ALL_PR, "Regional trade"); //it was exp in gams
    declareVariable("sa", DOM_PRI_PR,    "Supply to abroad (exports)");
    declareVariable("sc", DOM_PRI_PR,    "Supply, composite");
    declareVariable("sl", DOM_ALL_PR,    "Supply to locals");
    //declareVariable("st", DOM_PRI_PR,    "Supply, total", 0.0,UBOUND_MAX,"","in");
}
/**
Declare the constrains and their properties. For the domain type @see BaseClass
*/
void
Opt::declareConstrains (){
  // domain of constrains variables
  // for domain
  constrain mkeq2;
  mkeq2.name="mkeq2";
  mkeq2.comment="[h1] Conservation of matters of transformed products";
  mkeq2.domain=DOM_SEC_PR;
  mkeq2.direction = CONSTR_EQ;
  //mkeq2.evaluate = Opt::mkteq2f;

  constrain mkeq3;
  mkeq3.name="mkeq3";
  mkeq3.comment="[h2] Conservation of matters of raw products";
  mkeq3.domain=DOM_PRI_PR;
  mkeq3.direction = CONSTR_EQ;
  //mkeq3.evaluate = Opt::mkteq3f;

  constrain mkeq4;
  mkeq4.name="mkeq4";
  mkeq4.comment="[eq 13] Leontief transformation function";
  mkeq4.domain=DOM_PRI_PR;
  mkeq4.direction = CONSTR_EQ;

  constrain mkeq5;
  mkeq5.name="mkeq5";
  mkeq5.comment="[eq 21] Raw product supply function";
  mkeq5.domain=DOM_PRI_PR;
  mkeq5.direction = CONSTR_EQ;

  constrain mkeq6;
  mkeq6.name="mkeq6";
  mkeq6.comment="[eq 20] Trasformed products demand function";
  mkeq6.domain=DOM_SEC_PR;
  mkeq6.direction = CONSTR_EQ;

  constrain mkeq7;
  mkeq7.name="mkeq7";
  mkeq7.comment="[h7 and h3] Transformed products import function";
  mkeq7.domain=DOM_SEC_PR;
  mkeq7.direction = CONSTR_EQ;

  constrain mkeq8;
  mkeq8.name="mkeq8";
  mkeq8.comment="[h8 and h4] Raw products export function";
  mkeq8.domain=DOM_PRI_PR;
  mkeq8.direction = CONSTR_EQ;

  constrain mkeq13;
  mkeq13.name="mkeq13";
  mkeq13.comment="[h9] Calculation of the composite price of transformed products (PPC_Dp)";
  mkeq13.domain=DOM_SEC_PR;
  mkeq13.direction = CONSTR_EQ;

  constrain mkeq14;
  mkeq14.name="mkeq14";
  mkeq14.comment="[h10] Calculation of the composite price of raw products (PPC_Sw)";
  mkeq14.domain=DOM_PRI_PR;
  mkeq14.direction = CONSTR_EQ;

  constrain mkeq17;
  mkeq17.name="mkeq17";
  mkeq17.comment="[h16] Constrain of the transformaton supply (lower than the regional maximal production capacity)";
  mkeq17.domain=DOM_SEC_PR;
  mkeq17.direction = CONSTR_LE0;


  constrain mkeq23;
  mkeq23.name="mkeq23";
  mkeq23.comment="[h3] Composit demand eq. (Dp)";
  mkeq23.domain=DOM_SEC_PR;
  mkeq23.direction = CONSTR_EQ;

  constrain mkeq24;
  mkeq24.name="mkeq24";
  mkeq24.comment="[h4] Composite supply eq. (Sw)";
  mkeq24.domain=DOM_PRI_PR;
  mkeq24.direction = CONSTR_EQ;

  constrain mkeq26;
  mkeq26.name="mkeq26";
  mkeq26.comment="[eq ] Verification of the null transport agents supply";
  mkeq26.domain=DOM_R2_ALL_PR;
  mkeq26.direction = CONSTR_LE0;

  constrain mkeq25;
  mkeq25.name="mkeq25";
  mkeq25.comment="Verification of the null trasformers supply (price of raw product + trasf product > trasf product)";
  mkeq25.domain=DOM_SEC_PR;
  mkeq25.direction = CONSTR_GE0;

  constrain mkeq18;
  mkeq18.name="mkeq18";
  mkeq18.comment="Constrain on raw material supply (lower than inventory)";
  mkeq18.domain=DOM_PRI_PR;
  mkeq18.direction = CONSTR_LE0;

  constrain resbounds;
  resbounds.name="resbounds";
  resbounds.comment="Constrain on raw material supply (lower than inventory, for each possible combination of primary products)";
  resbounds.domain=DOM_PRI_PR_ALLCOMBS;
  resbounds.direction = CONSTR_LE0;



  //constrain steq;
  //steq.name="steq";
  //steq.comment="computation of total supply";
  //steq.domain=DOM_PRI_PR;
  //steq.direction = CONSTR_EQ;

  cons.push_back(mkeq2);
  cons.push_back(mkeq6);
  cons.push_back(mkeq7);
  cons.push_back(mkeq13);
  cons.push_back(mkeq23);
  cons.push_back(mkeq3);
  cons.push_back(mkeq4);
  cons.push_back(mkeq5);
  cons.push_back(mkeq8);
  cons.push_back(mkeq14);
  cons.push_back(mkeq24);
  cons.push_back(mkeq17);
  cons.push_back(mkeq26);
  cons.push_back(mkeq25);
  //cons.push_back(mkeq18);
  cons.push_back(resbounds);
  //cons.push_back(steq);
;



}
/**
Define the objective function
*/
template<class T> bool
Opt::eval_obj(Index n, const T *x, T& obj_value){

  double aa, bb, dc0, sigma, a_pr, ct, m, zeromax,supCorr2;
  obj_value = 0.;
  zeromax = 0.;

  for (uint r1=0;r1<l2r.size();r1++){
    for (uint r2=0;r2<l2r[r1].size();r2++){
    //  // consumer's surplus..
    //  sum ((i,p_tr),
    //    AA(i,p_tr)*(RVAR('dc',i,p_tr)**((sigma(p_tr)+1)/sigma(p_tr)))
    //    - AA(i,p_tr)*((0.5*dc0(i,p_tr))**((sigma(p_tr)+1)/sigma(p_tr)))
    //    - RVAR('pc',i,p_tr)*RVAR('dc',i,p_tr)
    //  )
```

```
    // 20161003: TODO: check if subsidies should enter also the obj function other than the bounds equations. For the moment, as agreed with Sylvain, they are left outside the obj function, but I am not sure of it.
      for (uint p=0;p<secPr.size();p++){
        aa = gpd("aa",l2r[r1][r2],secPr[p]);
        sigma = gpd("sigma",l2r[r1][r2],secPr[p]);
        dc0 = gpd("dc",l2r[r1][r2],secPr[p],secondYear);
        obj_value += aa*pow(mymax(zeromax,x[gix("dc",r1,r2,p)]),(sigma+1)/sigma)-aa*pow(mymax(zeromax,0.5*dc0),(sigma+1)/sigma)-x[gix("pc",r1,r2,p+nPriPr)]*x[gix("dc",r1,r2,p)];
      }
      //  // producers surplus..
      //  + sum((i,p_pr),
      //     RVAR(’pc’,i,p_pr)*RVAR(’sc’,i,p_pr)
      //     - BB(i,p_pr)*(RVAR(’sc’,i,p_pr)**((sigma(p_pr)+1)/sigma(p_pr)))
      //  )
      for (uint p=0;p<priPr.size();p++){
        bb = gpd("bb",l2r[r1][r2],priPr[p]);
        sigma = gpd("sigmaCorr",l2r[r1][r2],priPr[p]);
        //supCorr2 = gpd("supCorr2",l2r[r1][r2],priPr[p]);
        obj_value += x[gix("pc",r1,r2,p)]*x[gix("sc",r1,r2,p)] - bb*pow(mymax(zeromax,x[gix("sc",r1,r2,p)]),((sigma+1)/sigma));
      }
      //  // trasformations between primary products
      //  + sum ((i,p_pr,p_pr2),
      //  +RVAR(’pc’,i,p_pr2)*pres(p_pr,p_pr2)*RVAR(’sc’,i,p_pr)
      //  -BB(i,p_pr2)*(pres(p_pr,p_pr2)*RVAR(’sc’,i,p_pr))**((sigma(p_pr2)+1)/sigma(p_pr2))
      //   )

      for (uint p1=0;p1<priPr.size();p1++){
        for (uint p2=0;p2<priPr.size();p2++){
          a_pr = gpd("a_pr",l2r[r1][r2],priPr[p1],DATA_NOW,priPr[p2]);
          bb = gpd("bb",l2r[r1][r2],priPr[p2]);
          sigma = gpd("sigmaCorr",l2r[r1][r2],priPr[p2]);
          obj_value += x[gix("pc",r1,r2,p2)]*a_pr*x[gix("sc",r1,r2,p1)]-bb*pow(mymax(zeromax,a_pr*x[gix("sc",r1,r2,p1)]),(sigma+1)/sigma);
        }
      }
      //  // surplus of transport agents..
      //  + sum((i,j,prd), (RVAR(’pl’,j,prd)-RVAR(’pl’,i,prd)-CT(i,j,prd))*EXP(i,j,prd))
      for (uint p=0;p<allPr.size();p++){
        for (uint r2To=0;r2To<l2r[r1].size();r2To++){
          ct = gpd("ct",l2r[r1][r2],allPr[p],DATA_NOW,i2s(l2r[r1][r2To]));
          obj_value += (x[gix("pl",r1,r2To,p)]-x[gix("pl",r1,r2,p)]-ct)*x[gix("rt",r1,r2,p,r2To)];
        }
      }

      //  // transformers surplus..
      //  + sum((i,p_tr), (RVAR(’pl’,i,p_tr)-m(i,p_tr))*(RVAR(’sl’,i,p_tr))) // attenction it’s local. if we include w imports or p expports this have to change
      for (uint p=0;p<secPr.size();p++){
        m = gpd("m",l2r[r1][r2],secPr[p]);
        obj_value += (x[gix("pl",r1,r2,p+nPriPr)]-m)*x[gix("sl",r1,r2,p+nPriPr)];
      }
      //  - sum((i,p_pr), RVAR(’pl’,i,p_pr)*RVAR(’dl’,i,p_pr))                // to total and an other equation total=local+abroad should be added
      for (uint p=0;p<priPr.size();p++){
        obj_value -= x[gix("pl",r1,r2,p)]*x[gix("dl",r1,r2,p)];
      }
    } // end of each lev2 regions

  } //end of each r1 regions

  //obj_value = -obj_value; // we want maximisation, ipopt minimize! (donei n the options - scaling obj function)

 //exit(0);
  return true;
  // checked 20120802 this function is ok with gams, both in input and in output of the preoptimisation stage

}



/** Template function to implement (define) the previously declared constains.
To the initial macro loop it must be passed the product vector over where to loop (priPr, secPr or allPr) and the order of the constrain has it has been added to the const vector.
It could be possible to change this in a map and uses name, but then we would loose control on the constrains order, and we saw that it matters for finding the equilibrium.

*/
template<class T> bool
Opt::eval_constraints(Index n, const T *x, Index m, T* g){

  double a_pr, a, sigma, ff, sub_s, sub_d, sub_d_pSubstituted, sub_d_1, sub_d_1_pSubstituted, gg, q1, p1v, t1, r1v, psi, eta, pworld, ct, k, dispor, mv, in, in_1, supCorr, es_d, pc_1, pc_1_pSubstituted;
  Index cix = 0;
  Index debug = 0;

  // mkteq2(i,p_tr)..  RVAR(’dl’,i,p_tr)+sum(j,EXP(i,j,p_tr))  =e=  RVAR(’sl’,i,p_tr)+sum(b,EXP(b,i,p_tr)); // h1
  CONSTRAIN_START_LOOP(secPr, 0) // attention! you have to give the same order number as you inserted in the cons vector
    //g[cix] = x[gix("dl",r1,r2,psec)]-x[gix("sl",r1,r2,psec)]+x[gix("da",r1,r2,p)];
    g[cix] = x[gix("dl",r1,r2,psec)]-x[gix("sl",r1,r2,psec)];
    for (uint r2To=0;r2To<l2r[r1].size();r2To++){
      g[cix] += x[gix("rt",r1,r2,psec,r2To)]-x[gix("rt",r1,r2To,psec,r2)];
    }
  CONSTRAIN_END_LOOP

  // mkteq6(i,p_tr)..  RVAR(’dc’,i,p_tr)  =e=  GG(i,p_tr)*(RVAR(’pc’,i,p_tr)**sigma(p_tr)); // eq. 20 20160216: added sustitution elasticity in the demand
  // DEMAND EQUATION of transformed products
  CONSTRAIN_START_LOOP(secPr,1)
    gg      = gpd("gg",l2r[r1][r2],secPr[p]);
    sigma   = gpd("sigma",l2r[r1][r2],secPr[p]);
    pc_1    = gpd("pc",l2r[r1][r2],secPr[p],previousYear);
    sub_d   = gpd("sub_d",l2r[r1][r2],secPr[p]);              // subside this year
    sub_d_1 = gpd("sub_d",l2r[r1][r2],secPr[p],previousYear); // subside previous year
    g[cix]  = - gg*pow(x[gix("pc",r1,r2,psec)],sigma);
    for (uint p2=0;p2<secPr.size();p2++){
      es_d      = gpd("es_d",l2r[r1][r2],secPr[p],DATA_NOW,secPr[p2]);
      pc_1_pSubstituted    = gpd("pc",l2r[r1][r2],secPr[p2],previousYear);
      sub_d_pSubstituted   = gpd("pc",l2r[r1][r2],secPr[p2]);               // subside this year for the substitute product
      sub_d_1_pSubstituted = gpd("pc",l2r[r1][r2],secPr[p2],previousYear);  // subside last year for the substitute product

      g[cix] *= pow(
               (
                 ((x[gix("pc",r1,r2,psec)]+sub_d) / (x[gix("pc",r1,r2,priPr.size()+p2)]+sub_d_pSubstituted))
                    /
                 ((pc_1+sub_d_1) / (pc_1_pSubstituted+sub_d_1_pSubstituted))
               ), es_d
             );
    }
    //g[cix] = x[gix("dc",r1,r2,p)]-gg*pow(x[gix("pc",r1,r2,psec)],sigma); // original without substitution elasticity
    g[cix] += x[gix("dc",r1,r2,p)];
  CONSTRAIN_END_LOOP

  // mkteq7(i,p_tr)..  RVAR(’da’,i,p_tr)/RVAR(’dl’,i,p_tr)   =e= ((q1(i,p_tr)*RVAR(’pl’,i,p_tr))/(p1(i,p_tr)*PT_t(p_tr)))**psi(i,p_tr); // h7 and h3 ?
  CONSTRAIN_START_LOOP(secPr,2)
    q1 = gpd("q1",l2r[r1][r2],secPr[p]);
    p1v = 1-q1;
    psi = gpd("psi",l2r[r1][r2],secPr[p]);
    pworld = gpd("pl", worldCodeLev2,secPr[p]);
    g[cix] = x[gix("da",r1,r2,p)]/x[gix("dl",r1,r2,psec)] - pow((q1*x[gix("pl",r1,r2,psec)])/(p1v*pworld), psi);
  CONSTRAIN_END_LOOP

  // mkteq13(i,p_tr).. RVAR(’pc’,i,p_tr)*RVAR(’dc’,i,p_tr)   =e= RVAR(’dl’,i,p_tr)*RVAR(’pl’,i,p_tr)+RVAR(’da’,i,p_tr)*PT_t(p_tr);  // h9
  CONSTRAIN_START_LOOP(secPr,3)
    pworld = gpd("pl", worldCodeLev2,secPr[p]);
    g[cix] = x[gix("pc",r1,r2,psec)]*x[gix("dc",r1,r2,p)]-x[gix("dl",r1,r2,psec)]*x[gix("pl",r1,r2,psec)]-x[gix("da",r1,r2,p)]*pworld;
  CONSTRAIN_END_LOOP

  // mkteq23(i,p_tr).. RVAR(’dc’,i,p_tr)   =e= (q1(i,p_tr)*(RVAR(’da’,i,p_tr)**((psi(i,p_tr)-1)/psi(i,p_tr)))+ p1(i,p_tr)*(RVAR(’dl’,i,p_tr)**((psi(i,p_tr)-1)/psi(i,
  CONSTRAIN_START_LOOP(secPr,4)
    q1 = gpd("q1",l2r[r1][r2],secPr[p]);
    psi = gpd("psi",l2r[r1][r2],secPr[p]);
    p1v = 1-q1;
    g[cix] = x[gix("dc",r1,r2,p)] -
      pow(
            q1 * pow(x[gix("da",r1,r2,p)],(psi-1)/psi)
          + p1v * pow(x[gix("dl",r1,r2,psec)],(psi-1)/psi),
          psi/(psi-1)
      );
  CONSTRAIN_END_LOOP

  // mkteq3(i,p_pr)..  RVAR(’dl’,i,p_pr)+sum(j,EXP(i,j,p_pr))    =e=  RVAR(’sl’,i,p_pr)+sum(b,EXP(b,i,p_pr))+sum(p_pr2, pres(p_pr2,p_pr)* RVAR(’sl’,i,p_pr2)); // h2
  CONSTRAIN_START_LOOP(priPr,5)
    //g[cix] = x[gix("dl",r1,r2,p)]-x[gix("sl",r1,r2,p)]-x[gix("sa",r1,r2,p)];
    g[cix] = x[gix("dl",r1,r2,p)]-x[gix("sl",r1,r2,p)];
    for (uint r2To=0;r2To<l2r[r1].size();r2To++){
      g[cix] += x[gix("rt",r1,r2,p,r2To)]-x[gix("rt",r1,r2To,p,r2)];
    }
    for (uint p2=0;p2<priPr.size();p2++){
      a_pr = gpd("a_pr",l2r[r1][r2],priPr[p2],DATA_NOW,priPr[p]);
      g[cix] -= a_pr*x[gix("sl",r1,r2,p2)];
    }
  CONSTRAIN_END_LOOP

  //mkteq4(i,p_pr)..  RVAR(’dl’,i,p_pr)  =e=  sum(p_tr, a(p_pr,p_tr)*(RVAR(’sl’,i,p_tr))); // eq. 13
  CONSTRAIN_START_LOOP(priPr,6)
    g[cix] = x[gix("dl",r1,r2,p)];
    for (uint p2=0;p2<secPr.size();p2++){
      a = gpd("a",l2r[r1][r2],priPr[p],DATA_NOW,secPr[p2]);
      g[cix] -= a*x[gix("sl",r1,r2,p2+nPriPr)];
    }
  CONSTRAIN_END_LOOP

  // mkteq5(i,p_pr)..  RVAR(’sc’,i,p_pr)  =e=  FF(i,p_pr)*(RVAR(’pc’,i,p_pr)**sigma(p_pr)); // eq. 21
  // SUPPLY EQUATION OF PRIMARY PRODUCTS
  CONSTRAIN_START_LOOP(priPr,7)
    ff = gpd("ff",l2r[r1][r2],priPr[p]);
    sub_s = gpd("sub_s",l2r[r1][r2],priPr[p]);
    sigma = gpd("sigmaCorr",l2r[r1][r2],priPr[p]);
    //g[cix] = x[gix("sc",r1,r2,p)]-mymax(ff*pow(x[gix("pc",r1,r2,p)],sigma),0.001);
    g[cix] = x[gix("sc",r1,r2,p)]-ff*pow(x[gix("pc",r1,r2,p)]+sub_s,sigma);
    //g[cix] = x[gix("sc",r1,r2,p)]-ff*pow(x[gix("pc",r1,r2,p)],sigma-0.0001);
  CONSTRAIN_END_LOOP


  // mkteq8(i,p_pr)..  RVAR(’sa’,i,p_pr)/RVAR(’sl’,i,p_pr)  =e= ((t1(i,p_pr)*RVAR(’pl’,i,p_pr))/(r1(i,p_pr)*PT_t(p_pr)))**eta(i,p_pr); // h8 and h4 ?
  CONSTRAIN_START_LOOP(priPr,8)
    t1 = gpd("t1",l2r[r1][r2],priPr[p]);
    r1v = 1-t1;
    eta = gpd("eta",l2r[r1][r2],priPr[p]);
    pworld = gpd("pl", worldCodeLev2,priPr[p]);
    g[cix] = x[gix("sa",r1,r2,p)]/x[gix("sl",r1,r2,p)] - pow((t1*x[gix("pl",r1,r2,p)])/(r1v*pworld),eta);
  CONSTRAIN_END_LOOP

  // mkteq14(i,p_pr).. RVAR(’pc’,i,p_pr)*RVAR(’sc’,i,p_pr)  =e= RVAR(’sl’,i,p_pr)*RVAR(’pl’,i,p_pr)+RVAR(’sa’,i,p_pr)*PT_t(p_pr);  // h10
  CONSTRAIN_START_LOOP(priPr,9)
    pworld = gpd("pl", worldCodeLev2,priPr[p]);
    g[cix] = x[gix("pc",r1,r2,p)]*x[gix("sc",r1,r2,p)]-x[gix("sl",r1,r2,p)]*x[gix("pl",r1,r2,p)]-x[gix("sa",r1,r2,p)]*pworld;
  CONSTRAIN_END_LOOP

  //mkteq24(i,p_pr).. RVAR(’sc’,i,p_pr)  =e= (t1(i,p_pr)*(RVAR(’sa’,i,p_pr)**((eta(i,p_pr)-1)/eta(i,p_pr)))+ r1(i,p_pr)*(RVAR(’sl’,i,p_pr)**((eta(i,p_pr)-1)/eta(i,
  CONSTRAIN_START_LOOP(priPr,10)
    t1 = gpd("t1",l2r[r1][r2],priPr[p]);
    r1v = 1-t1;
    eta = gpd("eta",l2r[r1][r2],priPr[p]);
    g[cix] = x[gix("sc",r1,r2,p)] -
          pow(
             t1 * pow(x[gix("sa",r1,r2,p)],(eta-1)/eta)
          + r1v * pow(x[gix("sl",r1,r2,p)],(eta-1)/eta),
          eta/(eta-1)
        );
  CONSTRAIN_END_LOOP

  // mkteq17(i,p_tr).. RVAR(’sl’,i,p_tr)  =l=  Kt(i,p_tr);     // h16 in the presentation paper
  CONSTRAIN_START_LOOP(secPr,11)
    k = gpd("k",l2r[r1][r2],secPr[p]);
    g[cix] = x[gix("sl",r1,r2,p+nPriPr)]-k;
  CONSTRAIN_END_LOOP

  // mkeq26(i,prd,j).. RVAR(’pl’,j,prd)-RVAR(’pl’,i,prd)-CT(i,j,prd) =l= 0;
  CONSTRAIN_START_LOOP(allPr,12)
    for (uint r2To=0;r2To<l2r[r1].size();r2To++){
      cix = gix(12, r1, r2, p,r2To); // attenction we must redefine it, as we are now in a r2to loop
      ct = gpd("ct",l2r[r1][r2],allPr[p],DATA_NOW,i2s(l2r[r1][r2To]));
      g[cix] = (x[gix("pl",r1,r2To,p)]-x[gix("pl",r1,r2,p)]-ct);
    }
  CONSTRAIN_END_LOOP

  // mkteq25(i,p_tr).. sum(p_pr, a(p_pr,p_tr)*RVAR(’pl’,i,p_pr))+m(i,p_tr)  =g=  (RVAR(’pl’,i,p_tr)); // price of raw products + transf cost > trasf product
  CONSTRAIN_START_LOOP(secPr,13)
    mv = gpd("m",l2r[r1][r2],secPr[p]);
    g[cix] = mv - x[gix("pl",r1,r2,p+nPriPr)];
    for (uint p2=0;p2<priPr.size();p2++){
      a = gpd("a",l2r[r1][r2],priPr[p2],DATA_NOW,secPr[p]);
      g[cix] += a * x[gix("pl",r1,r2,p2)];
    }
  CONSTRAIN_END_LOOP

//  // mkteq18(i,p_pr).. RVAR(’sa’,i,p_pr)+RVAR(’sl’,i,p_pr)  =l=  dispor(i,p_pr); // total supply lower than the available stock
//  CONSTRAIN_START_LOOP(priPr,14)
//    in = gpd("in",l2r[r1][r2],priPr[p]);
//    double d1 = gix("sa",r1,r2,p);
//    double d2 = gix("sl",r1,r2,p);
//    g[cix] = x[gix("sa",r1,r2,p)]+x[gix("sl",r1,r2,p)]-in;
//  CONSTRAIN_END_LOOP

  // resbounds(i, p_pr_comb).. RVAR(’sa’,i,p_pr)+RVAR(’sl’,i,p_pr)  =l=  dispor(i,p_pr); // total supply lower than the available stock - FOR all combination subsets of ins
  CONSTRAIN_START_LOOP(priPrCombs,14)
    //ModelRegion* REG = MTHREAD->MD->getRegion(l2r[r1][r2]); // possibly slower
    //in = REG->inResByAnyCombination[p];
    in = ins[r1][r2][p];
    //if(p==0){
    //  in = 1.0; // workaround to lead -1<0 rather than 0<0 for the first (empty) subset - notneeded
    //}
    g[cix] = -in;
    for(uint i=0;i<priPrCombs[p].size();i++){
      g[cix] += x[gix("sa",r1,r2,priPrCombs[p][i])]+x[gix("sl",r1,r2,priPrCombs[p][i])];
    }
    g[cix] -= overharvestingAllowance; //0.02 don’t work always, expecially intermediate scnearios, 0.1 seems to work but produce a large artefact 20160219: made it a parameter

  CONSTRAIN_END_LOOP

  //CONSTRAIN_START_LOOP(priPr,15)
  //    g[cix] = x[gix("st",r1,r2,p)]-(x[gix("sl",r1,r2,p)]+x[gix("sa",r1,r2,p)]);
  //CONSTRAIN_END_LOOP

  return true;
}


//  ******** NOTHING TO DO BELOW THIS LINE ********************************

Opt::Opt(ThreadManager* MTHREAD_h){
  MTHREAD = MTHREAD_h;
  nVar    = 0;
  nCons   = 0;
  debugRunOnce = false;
  initOpt = true;
}

Opt::~Opt(){

}


bool
Opt::get_nlp_info(Index& n, Index& m, Index& nnz_jac_g,
                  Index& nnz_h_lag, IndexStyleEnum& index_style){


  if(initOpt){
    // does this initialisation code only once
    priPr = MTHREAD->MD->getStringVectorSetting("priProducts");
    secPr = MTHREAD->MD->getStringVectorSetting("secProducts");
    allPr = priPr;
    allPr.insert( allPr.end(), secPr.begin(), secPr.end() );
    nPriPr = priPr.size();
    nSecPr = secPr.size();
    nAllPr = allPr.size();
    std::vector<int> l1regIds =  MTHREAD->MD->getRegionIds(1, true);
    nL2r =  MTHREAD->MD->getRegionIds(2, true).size();
    firstYear = MTHREAD->MD->getIntSetting("initialYear");
    secondYear = firstYear+1;
    worldCodeLev2 = MTHREAD->MD->getIntSetting("worldCodeLev2");

    for(uint i=0;i<l1regIds.size();i++){
      std::vector<int> l2ChildrenIds;
      ModelRegion* l1Region = MTHREAD->MD->getRegion(l1regIds[i]);
      std::vector<ModelRegion*> l2Childrens = l1Region->getChildren(true);
      for(uint j=0;j<l2Childrens.size();j++){
        l2ChildrenIds.push_back(l2Childrens[j]->getRegId());
      }
      if(l2ChildrenIds.size()){
        l2r.push_back(l2ChildrenIds);
      }
    }

    // Create a vector with all possible combinations of primary products
    priPrCombs = MTHREAD->MD->createCombinationsVector(nPriPr);
    nPriPrCombs = priPrCombs.size();

    // put the variables and their domain in the vars map
    declareVariables();

    // declaring the contrains...
    declareConstrains();

    // calculate number of variables and constrains..
    calculateNumberVariablesConstrains();

    // cache initial positions (variables and constrains)..
    cacheInitialPosition();

    // cache initial positions (variables and constrains)..
    cachePositions();

    //tempDebug();

    //debugPrintParameters();

  } // finish initialisation things to be done only the first year

  previousYear = MTHREAD->SCD->getYear()-1; // this has to be done EVERY years !!

  n = nVar; // 300; // nVar;
  m = nCons; // 70; // nCons;

  overharvestingAllowance = MTHREAD->MD->getDoubleSetting("overharvestingAllowance", DATA_NOW);

  copyInventoryResourses();

  generate_tapes(n, m, nnz_jac_g, nnz_h_lag);

  //if(initOpt){
  //  calculateSparsityPatternJ();
  //  calculateSparsityPatternH();
    //tempDebug();
  //}

  // use the C style indexing (0-based)
  index_style = C_STYLE;

  initOpt=false;
  return true;
}

bool
Opt::get_bounds_info(Index n, Number* x_l, Number* x_u, Index m, Number* g_l, Number* g_u){

  // Set the bounds for the endogenous variables..
  for (Index i=0; i<n; i++) {
    x_l[i] =  getBoundByIndex(LBOUND,i);
    x_u[i] =  getBoundByIndex(UBOUND,i);
  }

  // Set the bounds for the constraints..
  for (Index i=0; i<m; i++) {
    int direction = getConstrainDirectionByIndex(i);
    switch (direction){
      case CONSTR_EQ:
        g_l[i] = 0.;
        g_u[i] = 0.;
        break;
      case CONSTR_LE0:
        g_l[i] = -2e19;
        g_u[i] = 0.;
        break;
      case CONSTR_GE0:
        g_l[i] = 0.;
        g_u[i] = 2e19;
        break;
    }
  }
  return true;
}

bool
Opt::get_starting_point(Index n, bool init_x, Number* x, bool init_z, Number* z_L, Number* z_U,
                        Index m, bool init_lambda, Number* lambda){

  // function checked on 20120724 on a subset of 3 regions and 4 products. All variables initial values are correctly those outputted by gams in 2006.
  //int thisYear = MTHREAD->SCD->getYear();
  //int initialOptYear = MTHREAD->MD->getIntSetting("initialOptYear");
  //if(thisYear != initialOptYear) return true;

  //msgOut(MSG_DEBUG,"Giving optimising variables previous years value as starting point");
  // Here, we assume we only have starting values for x, if you code
  // your own NLP, you can provide starting values for the others if
  // you wish.
  assert(init_x == true);
  assert(init_z == false);
  assert(init_lambda == false);

  VarMap::iterator viter;

  // fixing the starting points for each variable at the level of the previous years
  for (viter = vars.begin(); viter != vars.end(); ++viter) {
    //string debugs = viter->first;
    int vdomtype = viter->second.domain;
    if(vdomtype==DOM_PRI_PR){
      for(uint r1=0;r1<l2r.size();r1++){
        for(uint r2=0;r2<l2r[r1].size();r2++){
          for(uint p=0;p<priPr.size();p++){
            x[gix(viter->first,r1,r2,p)]= gpd(viter->first,l2r[r1][r2],priPr[p],previousYear);
          }
        }
      }
    } else if (vdomtype==DOM_SEC_PR) {
      for(uint r1=0;r1<l2r.size();r1++){
        for(uint r2=0;r2<l2r[r1].size();r2++){
          for(uint p=0;p<secPr.size();p++){
            x[gix(viter->first,r1,r2,p)]= gpd(viter->first,l2r[r1][r2],secPr[p],previousYear);
          }
        }
      }
    } else if (vdomtype==DOM_ALL_PR) {
      for(uint r1=0;r1<l2r.size();r1++){
        for(uint r2=0;r2<l2r[r1].size();r2++){
          for(uint p=0;p<allPr.size();p++){
            x[gix(viter->first,r1,r2,p)]= gpd(viter->first,l2r[r1][r2],allPr[p],previousYear);
          }
        }
      }
    } else if (vdomtype==DOM_R2_ALL_PR) {
      for(uint r1=0;r1<l2r.size();r1++){
        for(uint r2=0;r2<l2r[r1].size();r2++){
          for(uint p=0;p<allPr.size();p++){
            for(uint r2To=0;r2To<l2r[r1].size();r2To++){
              x[gix(viter->first,r1,r2,p,r2To)]= gpd(viter->first,l2r[r1][r2],allPr[p],previousYear,i2s(l2r[r1][r2To]));
            }
          }
        }
      }
    } else {
      msgOut(MSG_CRITICAL_ERROR,"Try to setting the initial value of a variable of unknow type ("+viter->first+")");
    }
  }

  //msgOut(MSG_DEBUG,"Finisced initial value assignments");

  return true;
}


void
Opt::finalize_solution(SolverReturn status,
                       Index n, const Number* x, const Number* z_L, const Number* z_U,
                       Index m, const Number* g, const Number* lambda,
                       Number obj_value, const IpoptData* ip_data, IpoptCalculatedQuantities* ip_cq){

  printf("\n\nObjective value\n");
  printf("f(x*) = %e\n", obj_value);

  // --> here is where to code the assignment of optimal values to to spd()

  VarMap::iterator viter;

  // fixing the starting points for each variable at the level of the previous years
  for (viter = vars.begin(); viter != vars.end(); ++viter) {
    //string debugs = viter->first;
    int vdomtype = viter->second.domain;
    if(vdomtype==DOM_PRI_PR){
      for(uint r1=0;r1<l2r.size();r1++){
        for(uint r2=0;r2<l2r[r1].size();r2++){
          for(uint p=0;p<priPr.size();p++){
            spd(x[gix(viter->first,r1,r2,p)],viter->first,l2r[r1][r2],priPr[p]);
          }
        }
      }
    } else if (vdomtype==DOM_SEC_PR) {
      for(uint r1=0;r1<l2r.size();r1++){
        for(uint r2=0;r2<l2r[r1].size();r2++){
          for(uint p=0;p<secPr.size();p++){
            spd(x[gix(viter->first,r1,r2,p)],viter->first,l2r[r1][r2],secPr[p]);

          }
        }
      }
    } else if (vdomtype==DOM_ALL_PR) {
      for(uint r1=0;r1<l2r.size();r1++){
        for(uint r2=0;r2<l2r[r1].size();r2++){
          for(uint p=0;p<allPr.size();p++){
            spd(x[gix(viter->first,r1,r2,p)],viter->first,l2r[r1][r2],allPr[p]);
          }
        }
      }
    } else if (vdomtype==DOM_R2_ALL_PR) {
      for(uint r1=0;r1<l2r.size();r1++){
        for(uint r2=0;r2<l2r[r1].size();r2++){
          for(uint p=0;p<allPr.size();p++){
            for(uint r2To=0;r2To<l2r[r1].size();r2To++){
              //if(x[gix(viter->first,r1,r2,p,r2To)] > 0){
              //  cout << l2r[r1][r2] << "\t" << allPr[p] << "\t" << l2r[r1][r2To] << "\t" << x[gix(viter->first,r1,r2,p,r2To)] << endl;
              //}
              spd(x[gix(viter->first,r1,r2,p,r2To)],viter->first,l2r[r1][r2],allPr[p],DATA_NOW,false,i2s(l2r[r1][r2To]));
            }
          }
        }
      }
    } else {
      msgOut(MSG_CRITICAL_ERROR,"Try to setting the solved value of a variable of unknow type ("+viter->first+")");
    }
  }

    // memory deallocation of ADOL-C variables
    delete[] x_lam;

    free(rind_g);
    free(cind_g);

    delete[] rind_L;
    delete[] cind_L;

    free(rind_L_total);
    free(cind_L_total);
    free(jacval);
    free(hessval);

    for (int i=0;i<n+m+1;i++) {
        free(HP_t[i]);
    }
    free(HP_t);

}

//***************************************************************************
//
//
//          Nothing has to be changed below this point !!
//
//
//***************************************************************************


bool
Opt::eval_f(Index n, const Number* x, bool new_x, Number& obj_value){
  eval_obj(n,x,obj_value);

  return true;
}

bool
Opt::eval_grad_f(Index n, const Number* x, bool new_x, Number* grad_f){

  gradient(tag_f,n,x,grad_f);

  return true;
}

bool
Opt::eval_g(Index n, const Number* x, bool new_x, Index m, Number* g){

  eval_constraints(n,x,m,g);

  return true;
}

bool
```

```
Opt::eval_jac_g(Index n, const Number *x, bool new_x,Index m, Index nele_jac,
                Index* iRow, Index *jCol, Number* values){
  if (values == NULL) {
    // return the structure of the jacobian

    for(Index idx=0; idx<nnz_jac; idx++)
      {
  iRow[idx] = rind_g[idx];
  jCol[idx] = cind_g[idx];
      }
  }
  else {
    // return the values of the jacobian of the constraints

    sparse_jac(tag_g, m, n, 1, x, &nnz_jac, &rind_g, &cind_g, &jacval, options_g);

    for(Index idx=0; idx<nnz_jac; idx++)
      {
  values[idx] = jacval[idx];

      }
  }
  return true;
}

bool
Opt::eval_h(Index n, const Number* x, bool new_x, Number obj_factor, Index m, const Number* lambda,
            bool new_lambda, Index nele_hess, Index* iRow, Index* jCol, Number* values){


  if (values == NULL) {
    // return the structure. This is a symmetric matrix, fill the lower left
    // triangle only.

    for(Index idx=0; idx<nnz_L; idx++)
      {
  iRow[idx] = rind_L[idx];
  jCol[idx] = cind_L[idx];
      }
  }
  else {
    // return the values. This is a symmetric matrix, fill the lower left
    // triangle only

    for(Index idx = 0; idx<n ; idx++)
      x_lam[idx] = x[idx];
    for(Index idx = 0; idx<m ; idx++)
      x_lam[n+idx] = lambda[idx];
    x_lam[n+m] = obj_factor;

    sparse_hess(tag_L, n+m+1, 1, x_lam, &nnz_L_total, &rind_L_total, &cind_L_total, &hessval, options_L);

    Index idx = 0;
    for(Index idx_total = 0; idx_total <nnz_L_total ; idx_total++)
      {
  if((rind_L_total[idx_total] < (unsigned int) n) && (cind_L_total[idx_total] < (unsigned int) n))
    {
      values[idx] = hessval[idx_total];
      idx++;
    }
      }
  }

  return true;

  //return false;
}


//***************    ADOL-C part ***********************************

void
Opt::generate_tapes(Index n, Index m, Index& nnz_jac_g, Index& nnz_h_lag){
    /// Copied from http://bocop.org/
    Number *xp    = new double[n];
    Number *lamp  = new double[m];
    Number *zl    = new double[m];
    Number *zu    = new double[m];

    adouble *xa   = new adouble[n];
    adouble *g    = new adouble[m];
    adouble *lam  = new adouble[m];
    adouble sig;
    adouble obj_value;

    double dummy;
//  double *jacval;

    int i,j,k,l,ii;

    x_lam   = new double[n+m+1];

//  cout << " Avant get_start" << endl;
    get_starting_point(n, 1, xp, 0, zl, zu, m, 0, lamp);
//   cout << " Apres get_start" << endl;

    //if(initOpt){ // that's funny, if I use this I get it slighly longer times, whatever I then use trace_off() or trace_off(1) (save to disk, seems unnecessary). If I use regenerated tapes I have also slighly inaccurate results.
    trace_on(tag_f);

    for(Index idx=0;idx<n;idx++)
        xa[idx] <<= xp[idx];

    eval_obj(n,xa,obj_value);

    obj_value >>= dummy;

    trace_off();

    trace_on(tag_g);

    for(Index idx=0;idx<n;idx++)
        xa[idx] <<= xp[idx];

    eval_constraints(n,xa,m,g);


    for(Index idx=0;idx<m;idx++)
        g[idx] >>= dummy;

    trace_off();

    trace_on(tag_L);

    for(Index idx=0;idx<n;idx++)
        xa[idx] <<= xp[idx];
    for(Index idx=0;idx<m;idx++)
        lam[idx] <<= 1.0;
    sig <<= 1.0;

    eval_obj(n,xa,obj_value);

    obj_value *= sig;
    eval_constraints(n,xa,m,g);

    for(Index idx=0;idx<m;idx++)
        obj_value += g[idx]*lam[idx];

    obj_value >>= dummy;

    trace_off();
    //} // end of if initOpt()



    rind_g = NULL;
    cind_g = NULL;

    options_g[0] = 0;          /* sparsity pattern by index domains (default) */
    options_g[1] = 0;          /*                         safe mode (default) */
    options_g[2] = -1;         /*                     &jacval is not computed */
    options_g[3] = 0;          /*                column compression (default) */

    jacval=NULL;

    sparse_jac(tag_g, m, n, 0, xp, &nnz_jac, &rind_g, &cind_g, &jacval, options_g);

    options_g[2] = 0;
    nnz_jac_g = nnz_jac;

    unsigned int  **JP_f=NULL;                /* compressed block row storage */
    unsigned int  **JP_g=NULL;                /* compressed block row storage */
    unsigned int  **HP_f=NULL;                /* compressed block row storage */
    unsigned int  **HP_g=NULL;                /* compressed block row storage */
    unsigned int  *HP_length=NULL;            /* length of arrays */
    unsigned int  *temp=NULL;                 /* help array */

    int ctrl_H;

    JP_f = (unsigned int **) malloc(sizeof(unsigned int*));
    JP_g = (unsigned int **) malloc(m*sizeof(unsigned int*));
    HP_f = (unsigned int **) malloc(n*sizeof(unsigned int*));
    HP_g = (unsigned int **) malloc(n*sizeof(unsigned int*));
    HP_t = (unsigned int **) malloc((n+m+1)*sizeof(unsigned int*));
    HP_length = (unsigned int *) malloc((n)*sizeof(unsigned int));
    ctrl_H = 0;

    hess_pat(tag_f, n, xp, HP_f, ctrl_H);

    indopro_forward_safe(tag_f, 1, n, xp, JP_f);
    indopro_forward_safe(tag_g, m, n, xp, JP_g);
    nonl_ind_forward_safe(tag_g, m, n, xp, HP_g);

    for (i=0;i<n;i++)
    {
        if (HP_f[i][0]+HP_g[i][0]!=0)
        {
            if (HP_f[i][0]==0) // number of non zeros in the i-th row
            {
                HP_t[i] = (unsigned int *) malloc((HP_g[i][0]+HPOFF)*sizeof(unsigned int));
                for(j=0;j<=(int) HP_g[i][0];j++)
                {
                    HP_t[i][j] = HP_g[i][j];
                }
                HP_length[i] = HP_g[i][0]+HPOFF;
            }
            else
            {
                if (HP_g[i][0]==0) // number of non zeros in the i-th row
                {
                    HP_t[i] = (unsigned int *) malloc((HP_f[i][0]+HPOFF)*sizeof(unsigned int));
                    for(j=0;j<=(int) HP_f[i][0];j++)
                    {
                        HP_t[i][j] = HP_f[i][j];
                    }
                    HP_length[i] = HP_f[i][0]+HPOFF;
                }
                else
                {
                    HP_t[i] = (unsigned int *) malloc((HP_f[i][0]+HP_g[i][0]+
HPOFF)*sizeof(unsigned int));
                    k = l = j = 1;
                    while ((k<=(int) HP_f[i][0]) && (l <= (int) HP_g[i][0]))
                    {
                        if (HP_f[i][k] < HP_g[i][l])
                        {
                            HP_t[i][j]=HP_f[i][k];
                            j++; k++;
                        }
                        else
                        {
                            if (HP_f[i][k] == HP_g[i][l])
                            {
                                HP_t[i][j]=HP_f[i][k];
                                l++;j++;k++;
                            }
                            else
                            {
                                HP_t[i][j]=HP_g[i][l];
                                j++;l++;
                            }
                        }
                    } // end while

                    // Fill the end of the vector if HP_g[i][0] < HP_f[i][0]
                    for(ii=k;ii<=(int) HP_f[i][0];ii++)
                    {
                        HP_t[i][j] = HP_f[i][ii];
                        j++;
                    }

                    // Fill the end of the vector if HP_f[i][0] < HP_g[i][0]
                    for(ii=l;ii<=(int) HP_g[i][0];ii++)
                    {
                        HP_t[i][j] = HP_g[i][ii];
                        j++;
                    }

                }
            }
            HP_t[i][0]=j-1; // set the first element with the number of non zeros in the i-th line
            HP_length[i] = HP_f[i][0]+HP_g[i][0]+HPOFF; // length of the i-th line
        }
        else
        {
            HP_t[i] = (unsigned int *) malloc((HPOFF+1)*sizeof(unsigned int));
            HP_t[i][0]=0;
            HP_length[i]=HPOFF;
        }

//     if (i==(int)n-1)
//     {
//       cout << " DISPLAY FINAL TIME HP : " << endl;
//       for (ii=0;ii<=(int)HP_length[i];ii++)
//         cout << " -------> HP[last][" << ii << "] = " << HP_t[i][ii] << endl;
//     }
    }

//   cout << " Avant les boucles" << endl;
//   cout << " m = " << m << endl;

    for (i=0;i<m;i++)
    {
//     cout << i << " --> nnz JP_g = " << JP_g[i][0]+1 << " -- ";
        HP_t[n+i] = (unsigned int *) malloc((JP_g[i][0]+1)*sizeof(unsigned int));
        HP_t[n+i][0]=JP_g[i][0];

//     cout << HP_t[n+i][0] << endl;

        for(j=1;j<= (int) JP_g[i][0];j++)
        {
            HP_t[n+i][j]=JP_g[i][j];
//       cout << " ---------> " << HP_t[n+i][j] << endl;
//       cout << " --> HP_length[" << JP_g[i][j] << "] = " << HP_length[JP_g[i][j]] << "  --  HP_t[" << JP_g[i][j] << "][0] = " << HP_t[JP_g[i][j]][0]+1 << endl;
            // We write the rows allocated in the previous "for" loop
            // If the memory allocated for the row is not big enough :
            if (HP_length[JP_g[i][j]] <= HP_t[JP_g[i][j]][0]+1) //! test avec "<=" (avant on avait "<" : bug, acces memoire non allouee)
            {
//         cout << " ---------> WARNING " << endl;
//         cout << " At index " << JP_g[i][j] << endl;


                // save a copy of existing vector elements :
                temp = (unsigned int *) malloc((HP_t[JP_g[i][j]][0]+1)*sizeof(unsigned int));
                for(l=0;l<=(int)HP_t[JP_g[i][j]][0];l++)
                {
                    temp[l] = HP_t[JP_g[i][j]][l]; //! valgrind : invalid read
//           cout << " -------> l = " << l << " -- " << temp[l] << endl;
                }

//         cout << " ----------->  DISPLAY " << endl;
//         for(l=0;l<=(int)HP_t[JP_g[i][j]][0];l++)
//         {
//           temp[l] = HP_t[JP_g[i][j]][l]; //! valgrind : invalid read & write
//           cout << " -------> HP[machin][" << l << "] = " << HP_t[JP_g[i][j]][l] << endl; //! valgrind : invalid read
//         }


                // Free existing row, and allocate more memory for it :
//         cout << " Avant free  --> pointeur = " <<HP_t[JP_g[i][j]]<< endl;
                unsigned int machin = JP_g[i][j];
                free(HP_t[machin]); // !Problem double free or corruption
//         cout << " Apres free --> pointeur = " <<HP_t[JP_g[i][j]]<< endl;

                HP_t[JP_g[i][j]] = (unsigned int *) malloc(2*HP_length[JP_g[i][j]]*sizeof(unsigned int));
                HP_length[JP_g[i][j]] = 2*HP_length[JP_g[i][j]];

                // Put back the values in this bigger vector :
                for(l=0;l<=(int)temp[0];l++)
                    HP_t[JP_g[i][j]][l] =temp[l];
                free(temp);

//         HP_t[JP_g[i][j]] = (unsigned int*) realloc (HP_t[JP_g[i][j]], 2*HP_length[JP_g[i][j]] * sizeof(unsigned int));
//         HP_length[JP_g[i][j]] = 2*HP_length[JP_g[i][j]];
            }
            HP_t[JP_g[i][j]][0] = HP_t[JP_g[i][j]][0]+1; // The size of the row is one greater than before
            HP_t[JP_g[i][j]][HP_t[JP_g[i][j]][0]] = i+n; // Now adding the element at the end //! valgrind : invalid write
        }
    }
//   cout << " Apres les boucles" << endl;

    for(j=1;j<= (int) JP_f[0][0];j++)
    {
        if (HP_length[JP_f[0][j]] <= HP_t[JP_f[0][j]][0]+1) //! test avec "<=" (pour etre coherent avec la remarque ci dessus, mais pas de cas test, a verifier)
        {
            temp = (unsigned int *) malloc((HP_t[JP_f[0][j]][0])*sizeof(unsigned int));
            for(l=0;l<=(int)HP_t[JP_f[0][j]][0];l++)
                temp[l] = HP_t[JP_f[0][j]][l];
            free(HP_t[JP_f[0][j]]);
            HP_t[JP_f[0][j]] = (unsigned int *) malloc(2*HP_length[JP_f[0][j]]*sizeof(unsigned int));
            HP_length[JP_f[0][j]] = 2*HP_length[JP_f[0][j]];
            for(l=0;l<=(int)temp[0];l++)
                HP_t[JP_f[0][j]][l] =temp[l];
            free(temp);
        }
        HP_t[JP_f[0][j]][0] = HP_t[JP_f[0][j]][0]+1;
        HP_t[JP_f[0][j]][HP_t[JP_f[0][j]][0]] = n+m;
    }

    HP_t[n+m] = (unsigned int *) malloc((JP_f[0][0]+2)*sizeof(unsigned int));
    HP_t[n+m][0]=JP_f[0][0]+1;
    for(j=1;j<= (int) JP_f[0][0];j++)
        HP_t[n+m][j]=JP_f[0][j];
    HP_t[n+m][JP_f[0][0]+1]=n+m;

    set_HP(tag_L,n+m+1,HP_t); // set sparsity pattern for the Hessian

    nnz_h_lag = 0;
    for (i=0;i<n;i++)
    {
        for (j=1;j<=(int) HP_t[i][0];j++)
            if ((int) HP_t[i][j] <= i)
                nnz_h_lag++;
        free(HP_f[i]);
        free(HP_g[i]);
    }
    nnz_L = nnz_h_lag;

    options_L[0] = 0;
    options_L[1] = 1;

    rind_L_total = NULL;
    cind_L_total = NULL;
    hessval = NULL;

    sparse_hess(tag_L, n+m+1, -1, xp, &nnz_L_total, &rind_L_total, &cind_L_total, &hessval, options_L)
;

    rind_L = new unsigned int[nnz_L];
    cind_L = new unsigned int[nnz_L];
    rind_L_total = (unsigned int*) malloc(nnz_L_total*sizeof(unsigned int)); //! test
    cind_L_total = (unsigned int*) malloc(nnz_L_total*sizeof(unsigned int)); //! test

    unsigned int ind = 0;

    for (int i=0;i<n;i++)
        for (unsigned int j=1;j<=HP_t[i][0];j++)
    {
        if (((int) HP_t[i][j]>=i) &&((int) HP_t[i][j]<n))
        {
            rind_L[ind] = i;
            cind_L[ind++] = HP_t[i][j];
        }
    }

    ind = 0;
    for (int i=0;i<n+m+1;i++)
        for (unsigned int j=1;j<=HP_t[i][0];j++)
    {
        if ((int) HP_t[i][j]>=i)
        {
            rind_L_total[ind] = i;
            cind_L_total[ind++] = HP_t[i][j];
        }
    }

    for (i=0;i<m;i++) {
        free(JP_g[i]);
    }

    free(JP_f[0]);
    free(JP_f);
    free(JP_g);
    free(HP_f);
    free(HP_g);
    free(HP_length);

    delete[] lam;
    delete[] g;
    delete[] xa;
    delete[] zu;
    delete[] zl;
    delete[] lamp;
    delete[] xp;

}
```

```


// ***************    FFSM OPT specific part ***********************************

const int
Opt::gip(const string &varName) const{ // get initial position
  map<string, int>::const_iterator p;
  p=initPos.find(varName);
  if(p != initPos.end()) {
    return p->second;
  }
  else {
    msgOut(MSG_CRITICAL_ERROR, "Asking the initial position in the concatenated array of a variable ("+varName+") that doesn't exist!");
    return 0;
  }
}

const int
Opt::gip(const int &cn) const { // get initial position
  return cInitPos.at(cn);
}

const int
Opt::gdt(const string &varName){ // get domain type
  VarMap::const_iterator p;
  p=vars.find(varName);
  if(p != vars.end()) {
    return p->second.domain;
  }
  else {
    msgOut(MSG_CRITICAL_ERROR, "Asking the domain type of a variable ("+varName+") that doesn't exist!");
    return 0;
  }
}

const int
Opt::gdt(const int &cn){ // get domain type
  return cons.at(cn).domain;
}

template<class T> const int
Opt::gix_uncached(const T &v_or_c, int r1Ix, int r2Ix, int prIx, int r2IxTo){

  // attenction, for computational reason we are not checking the call is within vectors limits!!!

  int dType = gdt(v_or_c);
  int othCountriesRegions = 0;
  int othCountriesRegions_r2case = 0;
  for (uint i=0;i<r1Ix;i++){
    othCountriesRegions += l2r[i].size();
  }
  for (uint i=0;i<r1Ix;i++){
    othCountriesRegions_r2case +=l2r[i].size()*l2r[i].size();
  }

  switch (dType){
    case DOM_PRI_PR:
      return gip(v_or_c)+(othCountriesRegions+r2Ix)*nPriPr+prIx;
    case DOM_SEC_PR:
      return gip(v_or_c)+(othCountriesRegions+r2Ix)*nSecPr+prIx;;
    case DOM_ALL_PR:
      return gip(v_or_c)+(othCountriesRegions+r2Ix)*nAllPr+prIx;
    case DOM_R2_PRI_PR:
      return gip(v_or_c)+(othCountriesRegions_r2case)*nAllPr+(r2Ix*nPriPr+prIx)*l2r[r1Ix].size()+r2IxTo;
    case DOM_R2_SEC_PR:
      return gip(v_or_c)+(othCountriesRegions_r2case)*nAllPr+(r2Ix*nSecPr+prIx)*l2r[r1Ix].size()+r2IxTo;
    case DOM_R2_ALL_PR:
      return gip(v_or_c)+(othCountriesRegions_r2case)*nAllPr+(r2Ix*nAllPr+prIx)*l2r[r1Ix].size()+r2IxTo; // new 20120814, looping r1,r2,p,r2to
      // initial position + (other countries region pairs + same country other regions from pair + regions to)* number of all products+product
      //return gip(v_or_c)+(othCountriesRegions_r2case+r2Ix*l2r[r1Ix].size()+r2IxTo)*nAllPr+prIx; // looping r1,r2,r2to,p
    case DOM_SCALAR:
      return gip(v_or_c);
    case DOM_PRI_PR_ALLCOMBS:
      return gip(v_or_c)+(othCountriesRegions+r2Ix)*nPriPrCombs+prIx;
  default:
    msgOut(MSG_CRITICAL_ERROR,"Try to calculate the position of a variable (or constrain) of unknow type.");
    return 0;
  }
}


```

```
const int
Opt::gix(const string &varName, const int& r1Ix, const int& r2Ix, const int& prIx, const int& r2IxTo) const{
  // attenction, for computational reasons we are not checking the call is within vectors limits!!!
  map <string, vector < vector < vector < vector <int> > > > >::const_iterator p;
  p=vpositions.find(varName);
  if(p != vpositions.end()) {
    return p->second[r1Ix][r2Ix][prIx][r2IxTo];
  }
  else {
    msgOut(MSG_CRITICAL_ERROR, "Asking the position of a variable ("+varName+") that doesn’t exist!");
    return 0;
  }
}

const int
Opt::gix(const int &cn, const int& r1Ix, const int& r2Ix, const int& prIx, const int& r2IxTo) const {
  return cpositions[cn][r1Ix][r2Ix][prIx][r2IxTo];
}

void
Opt::cacheInitialPosition(){
  int vInitialPosition = 0;
  int cInitialPosition = 0;
  VarMap::iterator viter;
  for (viter = vars.begin(); viter != vars.end(); ++viter) {
    initPos.insert(pair<string, int>(viter->first, vInitialPosition));
    initPos_rev.insert(pair<int, string>(vInitialPosition,viter->first));
    vInitialPosition += getDomainElements(viter->second.domain);
  }
  for (uint i=0;i<cons.size();i++){
    cInitPos.push_back(cInitialPosition);
    cInitialPosition += getDomainElements(cons[i].domain);
  }
}

void
Opt::cachePositions(){

  // variables..
  VarMap::iterator viter;
  for (viter = vars.begin(); viter != vars.end(); ++viter) {
    vpositions.insert(pair<string, vector < vector < vector < vector <int> > > > >(viter->first, buildPositionVector(viter->first, viter->second.domain)));
  }
  // constrains..
  for (uint i=0; i<cons.size();i++){
    cpositions.push_back(buildPositionVector(i, cons[i].domain));
  }

}

template<class T> vector < vector < vector < vector <int> > > >
Opt::buildPositionVector(const T &v_or_c, int dType){
  int pVectorSize;

  switch (dType){
    case DOM_PRI_PR:
      pVectorSize= priPr.size();
      break;
    case DOM_SEC_PR:
      pVectorSize= secPr.size();
      break;
    case DOM_ALL_PR:
      pVectorSize= allPr.size();
      break;
    case DOM_R2_PRI_PR:
      pVectorSize= priPr.size();
      break;
    case DOM_R2_SEC_PR:
      pVectorSize= secPr.size();
      break;
    case DOM_R2_ALL_PR:
      pVectorSize= allPr.size();
      break;
    case DOM_SCALAR:
      pVectorSize= allPr.size(); // it will simply fill the matrix all with the same value (the ip)
      break;
    case DOM_PRI_PR_ALLCOMBS:
      pVectorSize= priPrCombs.size();
      break;
    default:
      msgOut(MSG_CRITICAL_ERROR,"Try to build the position of a variable (or contrain) of unknow type.");
  }
```

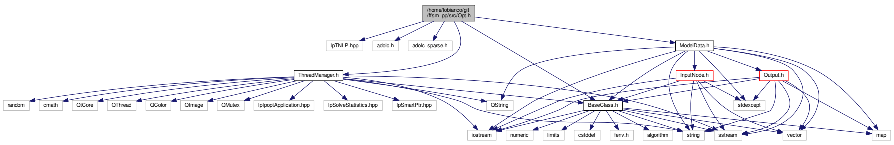
```


 vector < vector < vector < vector <int> > > > positionsToAdd;
 for(uint r1=0;r1<l2r.size();r1++){
  vector < vector < vector <int> > > dim1;
  for(uint r2=0;r2<l2r[r1].size();r2++){
   vector < vector <int> > dim2;
   for(uint p=0;p<pVectorSize;p++){
    vector <int> dim3;
     for(uint r2To=0;r2To<l2r[r1].size();r2To++){
      dim3.push_back(gix_uncached(v_or_c,r1,r2,p,r2To));
     }
    dim2.push_back(dim3);
   }
   dim1.push_back(dim2);
  }
  positionsToAdd.push_back(dim1);
 }
 return positionsToAdd;
}


void
Opt::calculateNumberVariablesConstrains(){
  // calculating the number of variables and the initial positions in the concatenated array..
  nVar = 0;
  VarMap::iterator viter;
  for (viter = vars.begin(); viter != vars.end(); ++viter) {
    nVar += getDomainElements(viter->second.domain);
  }

  // calculating the number of constrains..
  nCons = 0;
  nEqualityConstrains = 0;
  nLowerEqualZeroConstrains = 0;
  nGreaterEqualZeroConstrains = 0;
  for(uint i=0;i<cons.size();i++){
   nCons += getDomainElements(cons[i].domain);
   if(cons[i].direction == CONSTR_EQ){
    nEqualityConstrains += getDomainElements(cons[i].domain);
    continue;
   } else if (cons[i].direction == CONSTR_LE0) {
    nLowerEqualZeroConstrains += getDomainElements(cons[i].domain);
    continue;
   } else if (cons[i].direction == CONSTR_GE0) {
    nGreaterEqualZeroConstrains += getDomainElements(cons[i].domain);
    continue;
   } else {
    msgOut(MSG_CRITICAL_ERROR, "Asking for a constrain with unknown direction ("+i2s(cons[i].direction)+")");
   }
  }

  msgOut(MSG_INFO,"The model will work with "+i2s(nVar)+" variables and "+i2s(nCons)+" constrains ("+i2s(nEqualityConstrains)+" equalities, "+i2s(nLowerEqualZeroConstrains)+" lower than 0 and "+i2s(nGreaterEqualZeroConstrains)+" greater than 0)");
}

int
Opt::getDomainElements(int domain){
 int elements = 0;
 switch (domain){
  case DOM_PRI_PR:
   return nL2r*nPriPr;
  case DOM_SEC_PR:
   return nL2r*nSecPr;
  case DOM_ALL_PR:
   return nL2r*nAllPr;
  case DOM_R2_PRI_PR:
   for(uint r1=0;r1<l2r.size();r1++){
     elements += l2r[r1].size()*l2r[r1].size()*nPriPr; // EXP(i,j,p_pr)
   }
   return elements;
  case DOM_R2_SEC_PR:
   for(uint r1=0;r1<l2r.size();r1++){
     elements += l2r[r1].size()*l2r[r1].size()*nSecPr; // EXP(i,j,p_tr)
   }
   return elements;
  case DOM_R2_ALL_PR:
   for(uint r1=0;r1<l2r.size();r1++){
     elements += l2r[r1].size()*l2r[r1].size()*nAllPr; // EXP(i,j,prd)
   }
   return elements;
  case DOM_SCALAR:
   return 1;
  case DOM_PRI_PR_ALLCOMBS:
   return nL2r*nPriPrCombs;
  default:
    msgOut(MSG_CRITICAL_ERROR, "Asking for an unknown domain type ("+i2s(domain)+")");
  }
}

int
Opt::getConstrainDirectionByIndex(int idx){
  for(uint i=0;i<cons.size();i++){
    if(i!=cons.size()-1){
      if (idx >= gip(i) && idx < gip(i+1)){
        return cons[i].direction;
      }
    } else {
      if (idx >= gip(i) && idx < nCons){
      return cons[i].direction;
      }
    }
  }
  msgOut(MSG_CRITICAL_ERROR, "Asking contrain direction for an out of range contrain index!");
}

double
Opt::getBoundByIndex(const int & bound_type, const int & idx){

  map <int, string>::const_iterator p;
  p=initPos_rev.upper_bound(idx);
  p--;
  VarMap::const_iterator p2;
  p2=vars.find(p->second);
  if(p2 != vars.end()) {
    if (bound_type==LBOUND){
      if (p2->second.l_bound_var == ""){ // this var don't specific a variable as bound
        return p2->second.l_bound;
      } else {
        return getDetailedBoundByVarAndIndex(p2->second,idx,LBOUND);
      }
    } else if (bound_type==UBOUND){
      if (p2->second.u_bound_var == ""){ // this var don't specific a variable as bound
        return p2->second.u_bound;
      } else {
        return getDetailedBoundByVarAndIndex(p2->second,idx,UBOUND);
      }
    } else {
      msgOut(MSG_CRITICAL_ERROR, "Asking the bound with a type ("+i2s(bound_type)+") that I don't know how to handle !");
    }
  }
  else {
    msgOut(MSG_CRITICAL_ERROR, "Asking the bound from a variable ("+p->second+") that doesn't exist!");
  }
  return 0.;
}

double
Opt::getDetailedBoundByVarAndIndex(const endvar & var, const int & idx, const int & bType){
  // Tested 2015.01.08 with DOM_ALL_PR, DOM_PRI_PR, DOM_ALL_PR, DOM_R2_ALL_PR.
  int r1,r2,p,r2to;
  unpack(idx,var.domain,gip(var.name),r1,r2,p,r2to,true);
  //cout << "getBoundByVarAndIndex():\t" << var.name << '\t' << idx << '\t' << gip(var.name) << '\t' << r1 << '\t' << r2 << '\t' << p << '\t' << r2to << endl;
  //cout << "  --variables:\t" << var.l_bound_var << '\t' << var.u_bound_var << '\t' << "" << '\t' << l2r[r1][r2] << '\t' << "" << '\t' << allPr[p] << '\t' << l2r[r1][r2to] << endl;
  if(bType==LBOUND){
    if(r2to){
       return gpd(var.l_bound_var,l2r[r1][r2],allPr[p],DATA_NOW,i2s(l2r[r1][r2to]));
    } else {
      return gpd(var.l_bound_var,l2r[r1][r2],allPr[p],DATA_NOW,i2s(l2r[r1][r2to]));
    }
  } else {
    if(r2to){
       return gpd(var.u_bound_var,l2r[r1][r2],allPr[p]);
    } else {
      //cout << gpd(var.u_bound_var,l2r[r1][r2],allPr[p]) << endl;
      return gpd(var.u_bound_var,l2r[r1][r2],allPr[p]);
    }
  }
}

constrain*
Opt::getConstrainByIndex(int idx){
  for(uint i=0;i<cons.size();i++){
    if(i!=cons.size()-1){
      if (idx >= gip(i) && idx < gip(i+1)){
        return &cons[i];
      }
    } else {
      if (idx >= gip(i) && idx < nCons){
      return &cons[i];
      }
    }
  }
  msgOut(MSG_CRITICAL_ERROR, "Asking contrain direction for an out of range contrain index!");
}


void
Opt::unpack(int ix_h, int domain, int initial, int &r1_h, int &r2_h, int&p_h, int&r2to_h, bool fullp){
  ix_h = ix_h-initial;
  double ix=0;
  bool r2flag = false;
  int pIndexToAdd = 0;
  int np=0;
  if(domain==DOM_PRI_PR || domain==DOM_R2_PRI_PR) {
    np = nPriPr;
  } else if (domain==DOM_SEC_PR || domain==DOM_R2_SEC_PR) {
    np = nSecPr;
  } else if (domain==DOM_ALL_PR || domain==DOM_R2_ALL_PR) {
    np = nAllPr;
  } else if (domain=DOM_SCALAR){
    r1_h=0;r2_h=0;p_h=0;r2to_h=0;
    return;
  } else {
    msgOut(MSG_CRITICAL_ERROR,"unknow domain ("+i2s(domain)+") in unpack() function.");
  }
  if(domain==DOM_R2_PRI_PR || domain==DOM_R2_SEC_PR ||domain==DOM_R2_ALL_PR){
    r2flag = true;
  }
  if(fullp && (domain==DOM_SEC_PR || domain==DOM_R2_SEC_PR)){ // changed 20140107 (any how, previously the unpack() function was not used!!)
    pIndexToAdd = nPriPr;
    //cout << "pindexToAdd: " << pIndexToAdd << endl;
  }

  for (uint r1=0;r1<l2r.size();r1++){
    for (uint r2=0;r2<l2r[r1].size();r2++){
      for (uint p=0;p<np;p++){
        if(!r2flag){
          if(ix==ix_h){
            r1_h=r1;
            r2_h=r2;
            p_h=p+pIndexToAdd;
            r2to_h=0;
            return;
          }
          ix++;
        } else {
          for (uint r2To=0;r2To<l2r[r1].size();r2To++){
            if(ix==ix_h){
              r1_h=r1;
              r2_h=r2;
              p_h=p+pIndexToAdd;
              r2to_h=r2To;
              return;
            }
            ix++;
          }
        }
      }
    }
  }
  msgOut(MSG_CRITICAL_ERROR, "Error in unpack() function. Ix ("+i2s(ix_h)+") can not be unpacked");
}

int
Opt::getConNumber(constrain *con){
  for(uint i=0;i<cons.size();i++){
    if(   cons[i].name       == con->name
       && cons[i].comment    == con->comment
       && cons[i].domain     == con->domain
       && cons[i].direction  == con->direction){
      return i;
       }
  }
  msgOut(MSG_CRITICAL_ERROR,"Constrain didn’t found in list.");
}


```

```
void
Opt::calculateSparsityPatternJ(){

  unsigned int  **jacpat=NULL; // compressed row storage
  int           options_j[3]; // options for the jacobian patterns
  double        *x;
  int retv_j = -1; // return value

  options_j[0] = 0; // index domain propagation
  options_j[1] = 0; // automatic mode choice (ignored here)
  options_j[2] = 0; // safe
  jacpat = new unsigned int* [nCons];
  x = new double[nVar];

  nzjelements.clear();

  retv_j = jac_pat(tag_g, nCons, nVar,  x, jacpat, options_j);

  for (int i=0;i<nCons;i++) {
    for (int j=1;j<=jacpat[i][0];j++){
      vector <int> nzjelement;
      nzjelement.push_back(i);
      nzjelement.push_back(jacpat[i][j]);
      nzjelements.push_back(nzjelement);
    }
  }
}

void
Opt::calculateSparsityPatternH(){

  unsigned int  **hesspat=NULL; // compressed row storage
  int           options_h=0; // options for the hessian patterns
  double        *x;
  int retv_h = -1; // return value

  hesspat = new unsigned int* [(nVar+nCons+1)];
  x = new double[(nVar+nCons+1)];

  retv_h = hess_pat(tag_L,nVar+nCons+1,  x, hesspat, options_h);

  for (int i=0;i<(nVar);i++) {
    for (int j=1;j<=hesspat[i][0];j++){
      if(hesspat[i][j]<=i){
        vector <int> nzhelement;
        nzhelement.push_back(i);
        nzhelement.push_back(hesspat[i][j]);
        nzhelements.push_back(nzhelement);
      }
    }
  }
}

void
Opt::tempDebug(){

  cout << "Num of variables: " <<  nVar << " - Num of constrains:" << nCons << endl;
  cout << "IDX;ROW;COL" << endl;
  for(uint i=0;i<nzhelements.size();i++){
    cout << i << ";" << nzhelements[i][0] << ";" << nzhelements[i][1] << endl;
  }

  cout << "Dense jacobian: " << nCons * nVar << " elements" << endl;
  cout << "Dense hessian:  " << nVar*(nVar-1)/2+nVar << " elements" << endl;
  //exit(0);

}


const  Number&
Opt::mymax(const Number& a, const Number& b){
  return (a<b)?b:a;
}
const adouble&
Opt::mymax(const adouble& a, const adouble& b){
  return (a<b)?b:a;
}


bool
Opt::intermediate_callback(AlgorithmMode mode, Index iter, Number obj_value,
Number inf_pr, Number inf_du, Number mu, Number d_norm, Number regularization_size, Number alpha_du, Number
alpha_pr, Index ls_trials, const IpoptData *ip_data, IpoptCalculatedQuantities *ip_cq){
    int itnumber = iter;
    if(itnumber%10==0){
        msgOut(MSG_DEBUG,"Running ("+i2s(itnumber)+" iter) ..");
    }
    return true;
}

int
Opt::getVarInstances(const string& varName){
  return getDomainElements(gdt(varName));
}

/*
template <class T> const T&
Opt::mymax ( const T& a, const T& b ){
  return (a<b)?b:a;
}
*/
/**
 * @brief Opt::declareVariable
 * Define a single variable together with its domain and optionally its lower and upper bound (default 0.0, +inf)
 *
 * @param name    var name
 * @param domain  domain of the variable
 * @param l_bound lower bound (fixed)
 * @param u_bound upper bound (fixed)
 * @param l_bound_var variable name defining lower bound
 * @param u_bound_var variable name defining upper bound
 */

void
Opt::declareVariable(const string &name, const int & domain, const string &desc, const double & l_bound, const double & u_bound, const string & l_bound_var, const string & u_bound_var){
   endvar end_var;
   end_var.name = name;
   end_var.domain = domain;
   end_var.l_bound = l_bound;
   end_var.u_bound = u_bound;
   end_var.l_bound_var = l_bound_var;
   end_var.u_bound_var = u_bound_var;
   end_var.desc= desc;
   vars.insert(std::pair<std::string, endvar >(name, end_var));
}
/**
 * @brief Opt::createCombinationsVector
 * Return a vector containing any possible combination of nItems items (including all subsets).
 *
 * For example with nItems = 3:
 * 0: []; 1: [0]; 2: [1]; 3: [0,1]; 4: [2]; 5: [0,2]; 6: [1,2]; 7: [0,1,2]

 * @param nItems number of items to create p
 * @return A vector with in each slot the items present in that specific combination subset.
 */
/*
vector < vector <int> >
Opt::createCombinationsVector(const int& nItems) {
  // Not confuse combination with permutation where order matter. Here it doesn't matter, as much as the algorithm is the same and returns
  // to as each position always the same subset
  vector < vector <int> > toReturn;
  int nCombs = pow(2,nItems);
  //int nCombs = nItems;
  for (uint i=0; i<nCombs; i++){
    vector<int> thisCombItems; //concernedPriProducts;
    for(uint j=0;j<nItems;j++){
      uint j2 = pow(2,j);
      if(i & j2){ // bit a bit operator, p217 C++ book
        thisCombItems.push_back(j);
      }
    }
    toReturn.push_back(thisCombItems);
  }

//    cout << "N items:\t" << nItems << endl;
//    for (uint i=0;i<nCombs; i++){
//      cout << "  "<< i << ":\t";
//      for (uint j=0;j<toReturn[i].size();j++){
//        cout << toReturn[i][j] << "  ";
//      }
//      cout << endl;
//    }
//    exit(0);
  return toReturn;
}
*/

void
Opt::copyInventoryResourses(){
  // This function is not really needed, as actually the solver works also picking the region and the in dynamically
  // Caching the inventories in a vector should however be faster.
  // We now need it, as the vector inResByAnyCombination() account for the union between the inv set of the various pp. Also it now include the total mortality (alive plus death, if modelled)
  vector < vector < vector <double> > >  in_temp;
  for (uint r1=0;r1<l2r.size();r1++){
     vector < vector <double> > dim1;
    for (uint r2=0;r2<l2r[r1].size();r2++){
      vector <double> dim2;
      ModelRegion* REG = MTHREAD->MD->getRegion(l2r[r1][r2]);
      for (uint p=0;p<priPrCombs.size();p++){
        double this_in = REG->inResByAnyCombination[p];
        dim2.push_back(this_in);
      }
      dim1.push_back(dim2);
    }
    in_temp.push_back(dim1);
  }
  ins = in_temp;
}
```

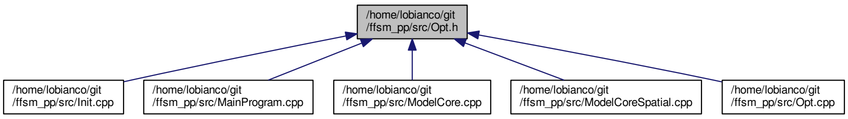
5.111 /home/lobianco/git/ffsm_pp/src/Opt.h File Reference

```
#include "IpTNLP.hpp"
#include <adolc.h>
#include <adolc_sparse.h>
#include "BaseClass.h"
#include "ThreadManager.h"
#include "ModelData.h"
```

Include dependency graph for Opt.h:



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



Classes

- class Opt
- struct constrain
- struct endvar

Macros

- #define tag_f 1
- #define tag_g 2
- #define tag_L 3
- #define HPOFF 30

5.111.1 Macro Definition Documentation

5.111.1.1 #define HPOFF 30

Definition at line 38 of file Opt.h.

Referenced by Opt::generate_tapes().

5.111.1.2 #define tag_f 1

Definition at line 35 of file Opt.h.

5.111.1.3 #define tag_g 2

Definition at line 36 of file Opt.h.

5.111.1.4 #define tag_L 3

Definition at line 37 of file Opt.h.

5.112 Opt.h

```
00001 /***************************************************************************
00002  *   Copyright (C) 2015 by Laboratoire d'Economie Forestière               *
00003  *   http://ffsm-project.org                                               *
00004  *                                                                         *
00005  *   This program is free software; you can redistribute it and/or modify  *
00006  *   it under the terms of the GNU General Public License as published by  *
00007  *   the Free Software Foundation; either version 3 of the License, or     *
00008  *   (at your option) any later version, given the compliance with the     *
00009  *   exceptions listed in the file COPYING that is distribued together     *
00010  *   with this file.                                                       *
00011  *                                                                         *
00012  *   This program is distributed in the hope that it will be useful,       *
00013  *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
00014  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
00015  *   GNU General Public License for more details.                          *
00016  *                                                                         *
00017  *   You should have received a copy of the GNU General Public License     *
00018  *   along with this program; if not, write to the                         *
00019  *   Free Software Foundation, Inc.,                                       *
00020  *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
00021  ***************************************************************************/
00022 #ifndef STDOPT_H
00023 #define STDOPT_H
00024 
00025 
00026 #include "IpTNLP.hpp"
00027 #include <adolc.h>
00028 #include <adolc_sparse.h>
00029 
00030 //regmas headers
00031 #include "BaseClass.h"
00032 #include "ThreadManager.h"
00033 #include "ModelData.h"
00034 
00035 #define tag_f 1
00036 #define tag_g 2
00037 #define tag_L 3
00038 #define HPOFF 30 //original: 30
00039 
00040 /// Class containing the optimization problem (the matrix and its methods)
00041 
00042 /**
00043 
00044 @author Antonello Lobianco
00045 */
00046 
00047 using namespace Ipopt;
```

```

struct constrain;
struct endvar;

class Opt: public BaseClass, public TNLP{
public:
            Opt(ThreadManager* MTHREAD_h); ///< Constructor
             ~Opt();
  /**@name Overloaded from TNLP */
  //@{
  /** Method to return some info about the nlp */
  virtual bool        get_nlp_info(Index& n, Index& m, Index& nnz_jac_g,
                   Index& nnz_h_lag, IndexStyleEnum& index_style);
  /** Method to return the bounds for my problem */
  virtual bool        get_bounds_info(Index n, Number* x_l, Number* x_u,
                    Index m, Number* g_l, Number* g_u);

  /** Method to return the starting point for the algorithm */
  virtual bool        get_starting_point(Index n, bool init_x, Number* x,
                       bool init_z, Number* z_L, Number* z_U,
                       Index m, bool init_lambda,
                       Number* lambda);
  /** Template to return the objective value */
  template<class T> bool eval_obj(Index n, const T *x, T& obj_value);

  /** Template to compute contraints */
  template<class T> bool eval_constraints(Index n, const T *x, Index m, T *g);

  /** Original method from Ipopt to return the objective value */
  /** remains unchanged */
  virtual bool        eval_f(Index n, const Number* x, bool new_x, Number& obj_value);

  /** Original method from Ipopt to return the gradient of the objective */
  /** remains unchanged */
  virtual bool        eval_grad_f(Index n, const Number* x, bool new_x, Number* grad_f);

  /**  Original method from Ipopt to return the constraint residuals */
  /** remains unchanged */
  virtual bool        eval_g(Index n, const Number* x, bool new_x, Index m, Number* g);

  /** Original method from Ipopt to return:
   *   1) The structure of the jacobian (if "values" is NULL)
  *   2) The values of the jacobian (if "values" is not NULL)
  */
  /** remains unchanged */
  virtual bool        eval_jac_g(Index n, const Number* x, bool new_x,
                   Index m, Index nele_jac, Index* iRow, Index *jCol,
                   Number* values);

  /** Original method from Ipopt to return:
  *   1) The structure of the hessian of the lagrangian (if "values" is NULL)
  *   2) The values of the hessian of the lagrangian (if "values" is not NULL)
  */
  /** remains unchanged */
  virtual bool        eval_h(Index n, const Number* x, bool new_x,
                 Number obj_factor, Index m, const Number* lambda,
                 bool new_lambda, Index nele_hess, Index* iRow,
                 Index* jCol, Number* values);

  //@}

  /** @name Solution Methods */
  //@{
  /** This method is called when the algorithm is complete so the TNLP can store/write the solution */
  virtual void finalize_solution(SolverReturn status,
                   Index n, const Number* x, const Number* z_L, const Number* z_U,
                   Index m, const Number* g, const Number* lambda,
                   Number obj_value,
                   const IpoptData* ip_data,
                   IpoptCalculatedQuantities* ip_cq);
  //@}

    /** Return information on each iteration */
    virtual bool intermediate_callback(AlgorithmMode mode,
                                       Index iter,
                                       Number obj_value,
                                       Number inf_pr,
                                       Number inf_du,
                                       Number mu,
                                       Number d_norm,
                                       Number regularization_size,
                                       Number alpha_du,
                                       Number alpha_pr,
                                       Index ls_trials,
                                       const IpoptData *ip_data,
                                       IpoptCalculatedQuantities *ip_cq);

  //***************    start ADOL-C part ***********************************

  /** Method to generate the required tapes */
  virtual void generate_tapes(Index n, Index m, Index& nnz_jac_g, Index& nnz_h_lag);

  //***************    end   ADOL-C part ***********************************

  // **************   start FFSM part ************************************
  void                declareVariables(); ///< declare the variables, their domains and their bounds
  void                declareVariable(const string &name, const int & domain, const string &desc= "", const double & l_bound=0.0, const double & u_bound=UBOUND_MAX, const string & l_bound_var="", const string & u_bound_var="");  ///< Declare a single variable, its domain and its bounds
  void                declareConstrains(); ///< declare the constrains, their domain, their direction and their associated evaluation function
  void                cacheInitialPosition(); ///< cache the initial positions of the variables and the constrains
  void                calculateNumberVariablesConstrains(); ///< calculate the number of variables and constrains
  void                cachePositions(); ///< cache the exact position index (initial+f(r1,r2,p,r2To) for each variable and constrain
  int                 getDomainElements(int domain); ///< return the number of elements of a domain
template<class T> vector < vector < vector < vector <int> > > > buildPositionVector(const T &v_or_c, int dType); ///< build the matrix of the positions for a given variable or contrain
  int                 getVarInstances(const string& varName); ///< return the number of instances of a variable, given his domain type
  ///< build the matrix of the positions for a given variable or contrain
  void                calculateSparsityPatternJ();
  void                calculateSparsityPatternH();


  const Number& mymax(const Number& a, const Number& b);
  const adouble& mymax(const adouble& a, const adouble& b);

  //template <class T> const T& mymax ( const T& a, const T& b );

  // **************   end FFSM part **************************************


protected:

  // convenient handles to equivalent ModelData functions..
  const double        gpd(const string &type_h, const int& regId_h, const string &prodId_h, const int& year=DATA_NOW, const string &freeDim_h="") const {return MTHREAD->MD->getProdData(type_h, regId_h, prodId_h, year, freeDim_h);};
  const double        gfd(const string &type_h, const int& regId_h, const string &forType_h, const string &diamClass_h, const int& year=DATA_NOW) const {return MTHREAD->MD->getForData(type_h, regId_h, forType_h, diamClass_h, year);};
  void                spd(const double& value_h, const string &type_h, const int& regId_h, const string &prodId_h, const int& year=DATA_NOW, const bool& allowCreate=false, const string &freeDim_h="") const {MTHREAD->MD->setProdData(value_h, type_h, regId_h, prodId_h, year, allowCreate, freeDim_h);};
  void                sfd(const double& value_h, const string &type_h, const int& regId_h, const string &forType_h, const string &diamClass_h, const int& year=DATA_NOW, const bool& allowCreate=false) const {MTHREAD->MD->setForData(value_h, type_h, regId_h, forType_h, diamClass_h, year, allowCreate);};
  bool                app(const string &prod_h, const string &forType_h, const string &dClass_h) const {return MTHREAD->MD->assessProdPossibility(prod_h, forType_h, dClass_h);};
  const int           gip(const string &varName) const; ///< Get the initial index position of a given variable in the concatenated array
  const int           gip(const int &cn) const; ///< Return the initial index position of a certain constrain
  template<class T> const int gix_uncached(const T &v_or_c, int r1Ix, int r2Ix, int prIx, int r2IxTo=0); ///< Get the index in the concatenated array gived a certain var name (string) or constrain index (int), the reg lev1 index, the reg lev2 index and the prod. index
  const int           gix(const string &varName, const int& r1Ix, const int& r2Ix, const int& prIx, const int& r2IxTo=0) const; ///< Get the index in the concatenated array gived a certain var name, the reg lev1 index, the reg lev2 index and the prod. index
  const int           gix(const int &cn, const int& r1Ix, const int& r2Ix, const int& prIx, const int& r2IxTo=0) const; ///< Get the index in the concatenated array gived a certain constrain, the reg lev1 index, the reg lev2 index and the prod. index
  const int           gdt(const string &varName); ///< Get the domain type of a given variable
  const int           gdt(const int &cn); ///< Get the domain type of a given constrain
  int                 getConstrainDirectionByIndex(int idx); ///< Return the direction of a given constrain
  double              getBoundByIndex(const int & bound_type, const int & idx);  ///< Return the bound of a given variable (by index)
  double              getDetailedBoundByVarAndIndex(const endvar & var, const int & idx, const int & bType); ///< Return the bound of a given variable given the variable and the required index. Called by getBoundByIndex().
  constrain*          getConstrainByIndex(int idx);
  void                unpack(int ix_h, int domain, int initial, int &r1_h, int &r2_h, int&p_h, int&r2to_h, bool fullp=false); ///< Return the dimensions given a certain index, domain type and initial position
  int                 getConNumber(constrain* con); ///< Return the position in the cons vector
  //vector < vector <int> > createCombinationsVector(const int& nItems); ///< Return a vector containing any possible combination of nItems items (including any possible subset). The returned vector has in each slot the items present in that specific combination.
  void                copyInventoryResourses(); ///< Copy the inventoried resources in the in vector for better performances

  void tempDebug();
```

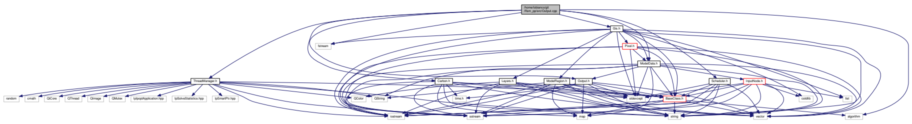
```
 //virtual void        eval_obj (Index n, const T *x, T& obj_value);

 vector<string>                           priPr;
 vector<string>                           secPr;
 vector<string>                           allPr;
 vector < vector <int> >                    l2r;
 vector < vector <int> >             priPrCombs; ///< A vector with all the possible combinations of primary products
 vector < vector < vector <double> > >      ins; ///< A copy of the inventoried resourses by region and primary product combination. It works also with dynamic loading of the region and the in, but it may be slower.
 map <string, int>                      initPos; ///< A map that returns the initial index position in the concatenated array for each variable
 map <int, string>                  initPos_rev; ///< A map with the name of the variable keyed by its initial position in the index
 vector<int>                           cInitPos; ///< A vector that returns the initial index position in the concatenated array for each constrain
 map <string, endvar>                      vars; ///< List of variables in the model and their domain: pr product, sec prod, all products or all products over each subregion pair (exports)
 map <string, vector < vector < vector < vector <int> > > > > vpositions; ///< cached position in the concatenated vector for each variables. Dimensions are l1reg, l2reg, prod, (l2To region).
 vector < vector < vector < vector < vector <int> > > > > cpositions; ///< cached position in the concatenated vector for each variables. Dimensions are contrain number, l1reg, l2reg, prod, (l2To region).
 int                                     nPriPr;
 int                                nPriPrCombs;
 int                                     nSecPr;
 int                                     nAllPr;
 int                                       nL2r;
 int                                       nVar;
 int                                      nCons;
 int                        nEqualityConstrains;
 int                  nLowerEqualZeroConstrains;
 int                nGreaterEqualZeroConstrains;
 int                               previousYear;
 int                                  firstYear;
 int                                 secondYear;
 int                              worldCodeLev2;
 bool                              debugRunOnce;
 double                 overharvestingAllowance; ///< Allows to harvest more than the resources available. Useful when resources got completelly exausted and the model refuses to solve.
 void                    debugPrintParameters();
 bool                                   initOpt;
 vector <constrain>                        cons;
 vector <vector <Index> >           nzjelements; ///< nzero elements for the jacobian matrix. nzelements[i][0] -> row (constrain), nzelements[i][1] -> column (variable)
 vector <vector <Index> >           nzhelements; ///< nzero elements for the hessian matrix


 /**@name Methods to block default compiler methods.
  * The compiler automatically generates the following three methods.
  *  Since the default compiler implementation is generally not what
  *  you want (for all but the most simple classes), we usually
  *  put the declarations of these methods in the private section
  *  and never implement them. This prevents the compiler from
  *  implementing an incorrect "default" behavior without us
  *  knowing. (See Scott Meyers book, "Effective C++")
  *
 */
 //@{
 //  MyADOLC_NLP();
 Opt(const Opt&);
 Opt& operator=(const Opt&);
 //@}

 //@{

 double *x_lam;

 //** variables for sparsity exploitation
 unsigned int **HP_t;         /* compressed block row storage */
 unsigned int *rind_g;        /* row indices    */
 unsigned int *cind_g;        /* column indices */
 double *jacval;              /* values         */
 unsigned int *rind_L;        /* row indices    */
 unsigned int *cind_L;        /* column indices */
 unsigned int *rind_L_total;  /* row indices    */
 unsigned int *cind_L_total;  /* column indices */
 double *hessval;             /* values */
 int nnz_jac;
 int nnz_L, nnz_L_total;
 int options_g[4];
 int options_L[4];


 //@}

```

```
};

struct constrain{
                          constrain(){comment="";};
  string                                    name;
  string                                 comment;
  int                                     domain;
  int                                  direction;

};

struct endvar{
  string                                    name;
  int                                     domain;
  string                                    desc; ///< Description of the variable
  double                                 l_bound; ///< A fixed numerical lower bound for all the domain
  double                                 u_bound; ///< A fixed numerical upper bound for all the domain
  string                             l_bound_var; ///< A variable giving the lower bound. If present, the value defined in the variable overrides l_bound.
  string                             u_bound_var; ///< A variable giving the upper bound. If present, the value defined in the variable overrides u_bound.
};



#endif

```

5.113 /home/lobianco/git/ffsm_pp/src/Output.cpp File Reference

```
#include <fstream>
#include <algorithm>
#include "Output.h"
#include "ThreadManager.h"
#include "Scheduler.h"
#include "ModelData.h"
#include "Gis.h"
#include "Carbon.h"
```

Include dependency graph for Output.cpp:



Typedefs

- typedef map< string, vector< double > > DataMap
- typedef pair< string, vector< double > > DataPair

5.113.1 Typedef Documentation

5.113.1.1 typedef map<string, vector <double> > DataMap

Definition at line 33 of file Output.cpp.

5.113.1.2 typedef pair<string, vector <double> > DataPair

Definition at line 34 of file Output.cpp.

5.114 Output.cpp

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d'Economie Forestière              *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#include <fstream>

#include <algorithm>

#include "Output.h"
#include "ThreadManager.h"
#include "Scheduler.h"
#include "ModelData.h"
#include "Gis.h"
#include "Carbon.h"

typedef map<string, vector <double> > DataMap;
typedef pair<string, vector <double> > DataPair;


Output::Output(ThreadManager* MTHREAD_h){
  MTHREAD=MTHREAD_h;
}

Output::~Output(){
}

// ---- functions ... ----------------------


void
Output::initOutput(){
  commonInit();
  initOutputMaps();
  initDebugOutput();
  initDebugPixelValues();
  initOutputForestData();
  initOutputProductData();
  initOptimisationLog();
  initCarbonBalance();
}


void
Output::commonInit(){
  oLevel       = MTHREAD->MD->getIntSetting("outputLevel");
  d            = getOutputFieldDelimiter();
  inYear       = MTHREAD->MD->getIntSetting("initialYear");
  nYears       = MTHREAD->MD->getIntSetting("simulationYears");
  baseDir      = MTHREAD->MD->getBaseDirectory();
  oDir         = MTHREAD->MD->getOutputDirectory();
//    bool initSeed = MTHREAD->MD->getBoolSetting("newRandomSeed");
//    if (initSeed){
//        uniform_int_distribution<> d(1, 1000000);
//        int random = d(*MTHREAD->gen);
//        scenarioName = MTHREAD->getScenarioName()+"_"+i2s(random);
//    } else {
//        scenarioName = MTHREAD->getScenarioName();
//    }
  if (MTHREAD->MD->getStringSetting("overridenScenarioName") == "none"){
    scenarioName = MTHREAD->getScenarioName();
  } else {
    scenarioName = MTHREAD->MD->getStringSetting("overridenScenarioName");
  }
  oFileExt     = MTHREAD->MD->getStringSetting("outputFileExtension");
  oHRedeable   = MTHREAD->MD->getBoolSetting("outputHumanReadable");
  oSingleFile  = MTHREAD->MD->getBoolSetting("outputSingleFile");
  oYears       = MTHREAD->MD->getIntVectorSetting("outYears");
  mapsOYears   = MTHREAD->MD->getIntVectorSetting("mapsOutYears");
  wRegId_l1    = MTHREAD->MD->getIntSetting("worldCodeLev1");
  wRegId_l2    = MTHREAD->MD->getIntSetting("worldCodeLev2");
  outForVariables          = MTHREAD->MD->
getStringVectorSetting("outForVariables");
  outProdVariables         = MTHREAD->MD->
getStringVectorSetting("outProdVariables");
  dClasses                 = MTHREAD->MD->
getStringVectorSetting("dClasses");
  pDClasses.insert(pDClasses.end(), dClasses.begin()+1,
dClasses.end() ); // production diameter classes
  dClasses.push_back(""); // needed for reporting of variables without diameter attribute
  outStepRange             = MTHREAD->MD->getIntSetting("outStepRange");
  forestDiamDetailedOutput = MTHREAD->MD->
getBoolSetting("forestDiamDetailedOutput");
  fTypes                   = MTHREAD->MD->getForTypeIds();

  priPr    = MTHREAD->MD->getStringVectorSetting("priProducts");
  secPr    = MTHREAD->MD->getStringVectorSetting("secProducts");
  allPr    = priPr;
  allPr.insert( allPr.end(), secPr.begin(), secPr.end() );
  nPriPr   = priPr.size();
  nSecPr   = secPr.size();
  nAllPr   = allPr.size();
  l1regIds = MTHREAD->MD->getRegionIds(1, true);
  nL2r     = MTHREAD->MD->getRegionIds(2, true).size();
  spMode   = MTHREAD->MD->getBoolSetting("usePixelData");
    //if(spMode) {
    //   pxIds =   getXyNPixels();
    //}


  for(uint i=0;i<l1regIds.size();i++){
    std::vector<int> l2ChildrenIds;
    ModelRegion* l1Region = MTHREAD->MD->getRegion(
l1regIds[i]);
    std::vector<ModelRegion*> l2Childrens = l1Region->getChildren(true);
    for(uint j=0;j<l2Childrens.size();j++){
      l2ChildrenIds.push_back(l2Childrens[j]->getRegId());
    }
    if(l2ChildrenIds.size()){
      l2r.push_back(l2ChildrenIds);
    }
  }

}

void
Output::initOptimisationLog(){
  if(oLevel<OUTVL_AGGREGATED) return;

    if (oSingleFile){
         logFilename = baseDir+oDir+"optimisationLogs/optimisationLogs.txt";

    } else {
         logFilename = baseDir+oDir+"optimisationLogs/"+
scenarioName+".txt";
    }


  ifstream in(logFilename.c_str(), ios::in);
  if(in.is_open()) { // file exist, no need to initializate it, but we are gonna clean it of previous data
 of the same scenario if present...
    in.close();
    cleanScenario(logFilename, scenarioName,
d);
    ofstream out(logFilename.c_str(), ios::app);
    if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+
logFilename+" for writing.");}
    time_t now;
    time(&now);
    struct tm *current = localtime(&now);
    string timemessage = i2s(current->tm_hour)+":"+i2s(current->tm_min)+":"+
i2s(current->tm_sec);
    out << scenarioName << d << "0000" << d << timemessage << d <<
d << d <<"\n";
    out.close();
    return;
  } else { // file doesn’t exist
    in.close();
    ofstream out(logFilename.c_str(), ios::out);
    if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+
logFilename+" for writing.");}
    out << "scenario" << d << "year" << d << "time" << d << "opt flag" << d << "iterations" <<
d <<"\n";
    time_t now;
    time(&now);
    struct tm *current = localtime(&now);
    string timemessage = i2s(current->tm_hour)+":"+i2s(current->tm_min)+":"+
i2s(current->tm_sec);
    out << scenarioName << d << "0000" << d << timemessage << d << d << d <<"\n";
    out.close();
  }
}

void
Output::initDebugOutput(){
    if(oLevel<OUTVL_ALL) return;

    // init debugging the expected returns...
    if(spMode) return;
    expReturnsDebugVariables.push_back("hVol_byUPp");
    expReturnsDebugVariables.push_back("hV_byFT");
    expReturnsDebugVariables.push_back("finalHarvestFlag");
    expReturnsDebugVariables.push_back("pondCoeff");
    expReturnsDebugVariables.push_back("pW");
    expReturnsDebugVariables.push_back("cumTp");
    expReturnsDebugVariables.push_back("vHa");
    expReturnsDebugVariables.push_back("expectedReturns");
    expReturnsDebugVariables.push_back("weightedAvgCompModeFlag");

    if (oSingleFile){
         debugFilename = baseDir+oDir+"debugs/debugOut.csv";
    } else {
         debugFilename = baseDir+oDir+"debugs/debugOut_"+
scenarioName+".csv";
    }

    ifstream in(debugFilename.c_str(), ios::in);
    if(in.is_open()) { // file exist, no need to initializate it, but we are gonna clean it of previous
 data of the same scenario if present...
        in.close();
        cleanScenario(debugFilename, scenarioName,
d);
        return;
    } else { // file doesn't exist
        in.close();
        ofstream out(debugFilename.c_str(), ios::out);
        if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+
debugFilename+" for writing.");}
        out << "scenario" << d << "year" << d << "region or pixel" << d << "forType" <<
d << "freeDim" << d << "prod" << d << "parName" << d << "value" << d <<"\n";
        out.close();
    }
}


void
Output::initDebugPixelValues(){
    if(oLevel<OUTVL_ALL) return;

    // init debugging the expected returns...
    if(!spMode) return;

    if (oSingleFile){
         debugPxValuesFilename = baseDir+oDir+"debugs/debugPxValues.csv";
    } else {
         debugPxValuesFilename = baseDir+oDir+"debugs/debugPxValues_"+
scenarioName+".csv";
    }

    ifstream in(debugPxValuesFilename.c_str(), ios::in);
    if(in.is_open()) { // file exist, no need to initializate it, but we are gonna clean it of previous
 data of the same scenario if present...
        in.close();
        cleanScenario(debugPxValuesFilename,
scenarioName, d);
        return;
    } else { // file doesn't exist
        in.close();
        ofstream out(debugPxValuesFilename.c_str(), ios::out);
        if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+
debugPxValuesFilename+" for writing.");}
        out << "scenario" << d << "year" << d << "region" << d << "pxId" << d << "pxX" <<
d << "pxY" << d ;
        for(uint f=0;f<fTypes.size();f++){
          string ft = fTypes[f];
          string header = "tp_multiplier_"+ft;
          out << header <<d;
        }
        for(uint f=0;f<fTypes.size();f++){
          string ft = fTypes[f];
          string header = "mortCoef_multiplier_"+ft;
          out << header <<d;
        }
        out << "var" << d ;

        for(uint f=0;f<fTypes.size();f++){
          string ft = fTypes[f];
          for (uint u=0;u<dClasses.size();u++){
            string dc=dClasses[u];
            string header = ft+"_"+dc;
            out << header <<d;
          }
        }
        out << "\n";


        out.close();
    }




    /*
    if(oSingleFile){
      outFileName = baseDir+oDir+"results/forestData"+oFileExt;
      ifstream in(outFileName.c_str(), ios::in);
      if(in.is_open()) { // file exist, no need to initializate it, but we are gonna clean it of previous data of the same scenario if present...
        in.close();
        cleanScenario(outFileName, scenarioName, d);
        return;
      } else {
        in.close();
      }
    } else {
      outFileName = baseDir+oDir+"results/forestData_"+scenarioName+oFileExt;
    }

    ofstream out(outFileName.c_str(), ios::out);
    if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+outFileName+" for reading.");}
      out << "scen" << d << "parName" << d << "country" << d << "region" << d << "forType" << d << "freeDim" << d;
    */









}

void
Output::initOutputForestData(){
  if(oLevel<OUTVL_DETAILED) return;

  if(oSingleFile){
    outFileName = baseDir+oDir+"results/forestData"+oFileExt;
    ifstream in(outFileName.c_str(), ios::in);
    if(in.is_open()) { // file exist, no need to initializate it, but we are gonna clean it of previous data of the same scenario if present...
      in.close();
      cleanScenario(outFileName, scenarioName, d);
      return;
    } else {
      in.close();
    }
  } else {
    outFileName = baseDir+oDir+"results/forestData_"+scenarioName+oFileExt;
  }

  ofstream out(outFileName.c_str(), ios::out);
  if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+outFileName+" for reading.");}
    out << "scen" << d << "parName" << d << "country" << d << "region" << d << "forType" << d << "freeDim" << d;
  if(oHRedeable){
    for(int i=0;i<nYears;i++){
      out << i+inYear << d;
    }
  } else {
    out << "year" << d << "value" << d;
  }
  out << "\n";
  out.close();
}

void
Output::initOutputProductData(){
  if(oLevel<OUTVL_DETAILED) return;

  if(oSingleFile){
    outFileName = baseDir+oDir+"results/productData"+oFileExt;
    ifstream in(outFileName.c_str(), ios::in);
    if(in.is_open()) { // file exist, no need to initializate it, but we are gonna clean it of previous data of the same scenario if present...
      in.close();
      cleanScenario(outFileName, scenarioName, d);
      return;
    } else {
      in.close();
    }
  } else {
    outFileName = baseDir+oDir+"results/productData_"+scenarioName+oFileExt;
  }

  ofstream out(outFileName.c_str(), ios::out);
  if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+outFileName+" for reading.");}
  out << "scen" << d << "parName" << d << "country" << d << "region" << d << "prod" << d << "freeDim" << d;
  if(oHRedeable){
    for(int i=0;i<nYears;i++){
      out << i+inYear << d;
    }
  } else {
    out << "year" << d << "value" << d;
  }
  out << "\n";
  out.close();
}

void
Output::initCarbonBalance(){

  if(oSingleFile){
    outFileName = baseDir+oDir+"results/carbonBalance"+oFileExt;
    ifstream in(outFileName.c_str(), ios::in);
    if(in.is_open()) { // file exist, no need to initializate it, but we are gonna clean it of previous data of the same scenario if present...
      in.close();
      cleanScenario(outFileName, scenarioName, d);
      return;
    } else {
      in.close();
    }
  } else {
    outFileName = baseDir+oDir+"results/carbonBalance_"+scenarioName+oFileExt;
  }

  ofstream out(outFileName.c_str(), ios::out);
  if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+outFileName+" for reading.");}
    out << "scen" << d << "country" << d << "region" << d << "balItem" << d;
  //if(oHRedeable){
  //  for(int i=0;i<nYears;i++){
  //    out << i+inYear << d;
  //  }
  //} else {
    out << "year" << d << "value" << d;
  //}
  out << "\n";
  out.close();
}


/**
Resetting the list of printed layers and the scenario name..
<br>Printing scenario name for post-processing scripts
*/
void
Output::initOutputMaps(){
  if(oLevel<OUTVL_MAPS) return;
  string mapBaseDirectory = baseDir+oDir+"maps/";
    string filenameToSaveScenarioName = mapBaseDirectory+"scenarioNames/"+
scenarioName;
    string filenameListIntLayers = mapBaseDirectory+"integerListLayers/"+scenarioName;
    string filenameListFloatLayers = mapBaseDirectory+"floatListLayers/"+scenarioName;

  // printing the scenario name in the "scenarioName file"...
  ofstream outSN(filenameToSaveScenarioName.c_str(), ios::out);
  if (!outSN){ msgOut(MSG_ERROR,"Error in opening the file "+filenameToSaveScenarioName+".");}
  outSN << scenarioName << "\n";
  outSN.close();
  // cleaning the "integerListLayers" and "floatListLayers" file...
  ofstream outi(filenameListIntLayers.c_str(), ios::out);
  outi.close();
  ofstream outf(filenameListFloatLayers.c_str(), ios::out);
  outf.close();
}

void
Output::print(){
  int cYear = MTHREAD->SCD->getYear();
  int initialSimulationYear = MTHREAD->MD->getIntSetting("initialOptYear");

  if (outStepRange != -1 && (cYear-initialSimulationYear)%outStepRange != 0 && cYear != (initialSimulationYear+nYears)-1 ) {
    cout << cYear << " not printed" << endl;
    return;
  }
  bool printThisYear = false;
  for(uint i=0;i<oYears.size();i++){
    if (outStepRange == -1 && oYears[i] == cYear) printThisYear = true;
  }
  if(outStepRange == -1 && !printThisYear) return;


  cout << "printing " << cYear << endl;
  printMaps();
  MTHREAD->MD->setErrorLevel(MSG_NO_MSG);
  printForestData(false);
  printProductData(false);
  printCarbonBalance();
  printDebugOutput();
  MTHREAD->MD->setErrorLevel(MSG_ERROR);
}

void
Output::printMaps(){
  if(oLevel<OUTVL_MAPS) return;
  int cYear = MTHREAD->SCD->getYear();
  if ( find(mapsOYears.begin(), mapsOYears.end(), cYear) != mapsOYears.end() ){
    MTHREAD->GIS->printLayers();
    if(oLevel<OUTVL_BINMAPS) return;
    MTHREAD->GIS->printBinMaps();
  }
}

void
Output::printFinalOutput(){
  // we do this only if we choosed the outputHumanReadable settings, as we flush the data all in ones at the end.
  // oterwise we flush data every year
  if(oHRedeable){
    MTHREAD->MD->setErrorLevel(MSG_NO_MSG);
    printForestData(true);
    printProductData(true);
    MTHREAD->MD->setErrorLevel(MSG_ERROR);
  }
}

void
Output::printForestData(bool finalFlush){

  if(oLevel<OUTVL_DETAILED) return;
  if(oHRedeable && !finalFlush) return;

  msgOut(MSG_INFO, "Printing forest data..");
  int currentYear = MTHREAD->SCD->getYear();
  if(oSingleFile){
    outFileName = baseDir+oDir+"results/forestData"+oFileExt;
  } else {
    outFileName = baseDir+oDir+"results/forestData_"+scenarioName+oFileExt;
  }
  ofstream out (outFileName.c_str(), ios::app);
```

```
 if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+outFileName+" for writing.");}
 double outvalue;
 for(uint v=0;v<outForVariables.size();v++){
  vector<string>fTypes_temp = fTypes;
  if( outForVariables[v]=="expReturns" || outForVariables[v]=="sumExpReturns" ||  outForVariables[v]=="totalShareInvadedArea") {
   fTypes_temp.push_back(""); // adding an empty forest type to report for variables that doesn't have a forestType dimension
   vector<string> ftParents = MTHREAD->MD->getForTypeParents();
   fTypes_temp.insert(fTypes_temp.end(),ftParents.begin(),ftParents.end()); // also inserting forest type "parents" for expected returns
  }
  for (uint r1=0;r1<l2r.size();r1++){
   for (uint r2=0;r2<l2r[r1].size();r2++){
    for(uint ft=0;ft<fTypes_temp.size();ft++){
     if(forestDiamDetailedOutput){
      for(uint dc=0;dc<dClasses.size();dc++){ // an empty "" dc has been already added to the vector
       out << scenarioName << d;
       out << outForVariables[v] << d;
       out << MTHREAD->MD->regId2RegSName(l1regIds.at(r1)) << d;
       out << MTHREAD->MD->regId2RegSName(l2r[r1][r2]) << d;
       out << fTypes_temp[ft] << d;
       out << dClasses[dc] << d;
       if (oHRedeable){
        for(int y=0;y<nYears;y++){
         outvalue = MTHREAD->MD->getForData(outForVariables[v],l2r[r1][r2],fTypes_temp[ft],dClasses[dc],y+inYear);
         out << outvalue << d;
        }
        out << "\n";
       } else {
        outvalue = MTHREAD->MD->getForData(outForVariables[v],l2r[r1][r2],fTypes_temp[ft],dClasses[dc]);
        out << currentYear << d;
        out << outvalue << d;
        out << "\n";
       }
      }
     } else {
      out << scenarioName << d;
      out << outForVariables[v] << d;
      out << MTHREAD->MD->regId2RegSName(l1regIds.at(r1)) << d;
      out << MTHREAD->MD->regId2RegSName(l2r[r1][r2]) << d;
      out << fTypes_temp[ft] << d;
      out << d;
      if (oHRedeable){
       for(int y=0;y<nYears;y++){
        outvalue = MTHREAD->MD->getForData(outForVariables[v],l2r[r1][r2],fTypes_temp[ft],DIAM_ALL,y+inYear);
        out << outvalue << d;
       }
       out << "\n";
      } else {
       outvalue = MTHREAD->MD->getForData(outForVariables[v],l2r[r1][r2],fTypes_temp[ft],DIAM_ALL);
       out << currentYear << d;
       out << outvalue << d;
       out << "\n";
      }
     }
    }
   }
  }
 }
 /*
 DataMap::const_iterator i;
 string key;
 vector <double> values;
 string parName;
 int regId;
 string forType;
 string diamClass;
 for(i=MTHREAD->MD->forDataMap.begin();i!=MTHREAD->MD->forDataMap.end();i++){
  key = i->first;
  values = i->second;
  MTHREAD->MD->unpackKeyForData(key, parName, regId, forType, diamClass);
  ModelRegion* REG = MTHREAD->MD->getRegion(regId);
  // we don't want to output data from residual region unless it's the world region we are speaking of
  if(REG->getIsResidual() && !(regId==wRegId_l1 || regId==wRegId_l2)) continue;
  out << scenarioName << d;
  out << parName << d;
```

```
    if (REG->getRegLevel()==2){
      ModelRegion* pREG = MTHREAD->MD->getRegion(REG->getParRegId());
      out << pREG->getRegSName() << d;
      out << REG->getRegSName() << d;
    } else if (REG->getRegLevel()==1){
      out << REG->getRegSName() << d;
      out << d;
    } else {
      out << d << d;
    }
    out << forType << d;
    out << diamClass << d;
    if (oHRedeable){
      for(int y=0;y<nYears;y++){
        out << MTHREAD->MD->getTimedData(values,y+inYear) << d;
      }
      out << "\n";
    } else {
      out << currentYear << d;
      out << MTHREAD->MD->getTimedData(values,currentYear) << d;
      out << "\n";
    }
  }
  */
  out.close();
}

void
Output::printProductData(bool finalFlush){

  if(oLevel<OUTVL_DETAILED) return;
  if(oHRedeable && !finalFlush) return;

  msgOut(MSG_INFO, "Printing market data..");
  int currentYear = MTHREAD->SCD->getYear();

  if(oSingleFile){
    outFileName = baseDir+oDir+"results/productData"+oFileExt;
  } else {
    outFileName = baseDir+oDir+"results/productData_"+scenarioName+oFileExt;
  }
  ofstream out (outFileName.c_str(), ios::app);
  if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+outFileName+" for writing.");}


  //11042  hardWSawnW  11083  0.00230651
  //11042  hardWSawnW  11082  0.0390874

  //if(MTHREAD->SCD->getYear() == 2007){
//  double test  = MTHREAD->MD->getProdData("rt",11042,"hardWSawnW",DATA_NOW);
//  double test2 = MTHREAD->MD->getProdData("rt",11042,"hardWSawnW",DATA_NOW,"11083");
//  double test3 = MTHREAD->MD->getProdData("rt",11042,"hardWSawnW",DATA_NOW,"11082");
//  cout << test << '\t' << test2 << '\t' << test3 << endl;
//  exit(0);
//  }

  double outvalue;
  for(uint v=0;v<outProdVariables.size();v++){
    for (uint r1=0;r1<l2r.size();r1++){
      for (uint r2=0;r2<l2r[r1].size();r2++){
        for(uint p=0;p<allPr.size();p++){

          if(outProdVariables[v]=="rt"){
            for(uint r2b=0;r2b<l2r[r1].size();r2b++){
              out << scenarioName << d;
              out << outProdVariables[v] << d;
              out << MTHREAD->MD->regId2RegSName(l1regIds.at(r1)) << d;
              out << MTHREAD->MD->regId2RegSName(l2r[r1][r2]) << d;
              out << allPr[p] << d;
              out << l2r[r1][r2b] << d;
              if (oHRedeable){
                for(int y=0;y<nYears;y++){
                  outvalue = MTHREAD->MD->getProdData(outProdVariables[v],l2r[r1][r2],allPr[p],y+inYear,i2s(l2r[r1][r2b]));
                  out << outvalue << d;
                }
                out << "\n";
              } else {
//                if(MTHREAD->SCD->getYear() == 2007 && l2r[r1][r2] == 11042 && allPr[p] == "hardWSawnW" && (l2r[r1][r2b]== 11083 || l2r[r1][r2b]== 11082 )){
//                outvalue = MTHREAD->MD->getProdData(outProdVariables[v],l2r[r1][r2],allPr[p],currentYear,i2s(l2r[r1][r2b]));
//                cout << outvalue << endl;
//                }
                outvalue = MTHREAD->MD->getProdData(outProdVariables[v],l2r[r1][r2],allPr[p],currentYear,i2s(l2r[r1][r2b]));
                out << currentYear << d;
                out << outvalue << d;
                out << "\n";
              }
            }
          } else {
            out << scenarioName << d;
            out << outProdVariables[v] << d;
            out << MTHREAD->MD->regId2RegSName(l1regIds.at(r1)) << d;
            out << MTHREAD->MD->regId2RegSName(l2r[r1][r2]) << d;
            out << allPr[p] << d;
            out << d;
            if (oHRedeable){
              for(int y=0;y<nYears;y++){
                outvalue = MTHREAD->MD->getProdData(outProdVariables[v],l2r[r1][r2],allPr[p],y+inYear);
                out << outvalue << d;
              }
              out << "\n";
            } else {
              outvalue = MTHREAD->MD->getProdData(outProdVariables[v],l2r[r1][r2],allPr[p]);
              out << currentYear << d;
              out << outvalue << d;
              out << "\n";
            }

          }
        }
      }
    }
  }




/*
  DataMap::const_iterator i;
  string key;
  vector <double> values;
  string parName;
  int regId;
  string prod;
  string freeDim;
  for(i=MTHREAD->MD->prodDataMap.begin();i!=MTHREAD->MD->prodDataMap.end();i++){
    key = i->first;
    values = i->second;
    MTHREAD->MD->unpackKeyProdData(key, parName, regId, prod, freeDim);
    ModelRegion* REG = MTHREAD->MD->getRegion(regId);
    // we don't want to output data from residual region unless it's the world region we are speaking of
    if(REG->getIsResidual() && !(regId==wRegId_l1 || regId==wRegId_l2)) continue;
    out << scenarioName << d;
    out << parName << d;
    if (REG->getRegLevel()==2){
      ModelRegion* pREG = MTHREAD->MD->getRegion(REG->getParRegId());
      out << pREG->getRegSName() << d;
      out << REG->getRegSName() << d;
    } else if (REG->getRegLevel()==1){
      out << REG->getRegSName() << d;
      out << d;
    } else {
      out << d << d;
    }
    out << prod << d;
    out << freeDim << d;
    if (oHRedeable){
      for(int y=0;y<nYears;y++){
        out << MTHREAD->MD->getTimedData(values,y+inYear) << d;
      }
      out << "\n";
    } else {
      out << currentYear << d;
      out << MTHREAD->MD->getTimedData(values,currentYear) << d;
      out << "\n";
    }
  }

*/
  out.close();
}

```

```
void
Output::printCarbonBalance(){

  int currentYear = MTHREAD->SCD->getYear();
  if (currentYear == inYear) {return;} // don’t print carbon balance on first year, carbon balance containers has not yet been initialised

  msgOut(MSG_INFO, "Printing forest data..");

  if(oSingleFile){
    outFileName = baseDir+oDir+"results/carbonBalance"+oFileExt;
  } else {
    outFileName = baseDir+oDir+"results/carbonBalance_"+scenarioName+oFileExt;
  }
  ofstream out (outFileName.c_str(), ios::app);
  if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+outFileName+" for writing.");}
  double outvalue=0;

  vector<int> balItems {STOCK_INV,STOCK_EXTRA,STOCK_PRODUCTS,EM_ENSUB,EM_MATSUB,EM_FOROP};

  for (uint r1=0;r1<l2r.size();r1++){
    for (uint r2=0;r2<l2r[r1].size();r2++){
      int regId = l2r[r1][r2];
      for (uint b=0;b<balItems.size();b++){
        out << scenarioName << d;
        out << MTHREAD->MD->regId2RegSName(l1regIds.at(r1)) << d;
        out << MTHREAD->MD->regId2RegSName(l2r[r1][r2]) << d;
        string balItemString;
        switch(balItems[b]){
          case STOCK_INV: {
            balItemString = "STOCK_INV";
            outvalue = MTHREAD->CBAL->getStock(regId, balItems[b]);
            break;
          }
          case STOCK_EXTRA: {
            balItemString = "STOCK_EXTRA";
            outvalue = MTHREAD->CBAL->getStock(regId, balItems[b]);
            break;
          }
          case STOCK_PRODUCTS: {
            balItemString = "STOCK_PRODUCTS";
            outvalue = MTHREAD->CBAL->getStock(regId, balItems[b]);
            break;
          }
          case EM_ENSUB: {
            balItemString = "EM_ENSUB";
            outvalue = MTHREAD->CBAL->getCumSavedEmissions(regId, balItems[b]);
            break;
          }
          case EM_MATSUB: {
            balItemString = "EM_MATSUB";
            outvalue = MTHREAD->CBAL->getCumSavedEmissions(regId, balItems[b]);
            break;
          }
          case EM_FOROP: {
            balItemString = "EM_FOROP";
            outvalue = MTHREAD->CBAL->getCumSavedEmissions(regId, balItems[b]);
            break;
          }
        default:
            msgOut(MSG_CRITICAL_ERROR,"Unexpected balance item type in function printCarbonBalance");
        }
        out << balItemString << d;
        out << currentYear << d;
        out << outvalue << d;
        out << "\n";


      } // end bal items
    } // end r2
  } // end r1
  out.close();
}
```

```
char
Output::getOutputFieldDelimiter(){
  int delimiterID = MTHREAD->MD->getIntSetting("outputFieldDelimiter");
  switch (delimiterID) {
    case 1:
      return ',';
      break;
    case 2:
      return ';';
      break;
    case 3:
      return ':';
      break;
    case 4:
      return '\t';
      break;
    case 5:
      return ' ';
      break;
    default:
      msgOut(MSG_ERROR, "You have specified an unknow output file field delimiter. Using \";\".");
      return ',';
  }
}

void
Output::printOptLog(bool optimal, int &nIterations, double &obj){
  if(oLevel<OUTVL_AGGREGATED) return;

  ofstream out(logFilename.c_str(), ios::app);
  if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+logFilename+" for writing.");}
  time_t now;
  time(&now);
  struct tm *current = localtime(&now);
  string timemessage = i2s(current->tm_hour)+":"+i2s(current->tm_min)+":"+i2s(current->tm_sec);
  out << scenarioName << d << MTHREAD->SCD->getYear() << d << timemessage << d << optimal;
  out << d << nIterations << d << obj << "\n";
  out.close();

}

void
Output::printDebugOutput(){
  if(oLevel<OUTVL_ALL) return;

  // print debugging the expected returns...

  if (!spMode && !expReturnsDebug.empty()){
    ofstream out (debugFilename.c_str(), ios::app);
    if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+debugFilename+" for writing.");}
    int currentYear = MTHREAD->SCD->getYear();
    vector <int> regIds2 = MTHREAD->MD->getRegionIds(2);

        for (uint r2=0;r2<regIds2.size();r2++){
            for(uint ft=0;ft<fTypes.size();ft++){
                for(uint dc=0;dc<(dClasses.size()-1);dc++){
                    for(uint pp=0;pp<priPr.size();pp++){
                        for(uint dv=0;dv<expReturnsDebugVariables.size();dv++){
                          // vector <vector < vector <vector <vector <double> > > > > expReturnsDebug;
                          double outputValue = expReturnsDebug.at(r2).at(ft).at(dc).at(pp).at(dv);
                          out << scenarioName << d;
                          out << currentYear << d;
                          out << MTHREAD->MD->regId2RegSName(regIds2[r2]) << d;
                          out << fTypes[ft] << d;
                          out << dClasses[dc] << d;
                          out << priPr[pp] << d;
                          out << expReturnsDebugVariables[dv] << d;
                          out << outputValue << d;
                          out << "\n";
                        }
                    }
                }
            }
        }

  } // end initial condition checks
}
```

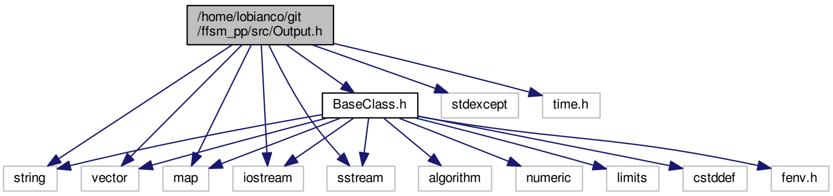
```
void
Output::printDebugPixelValues(){

  if(oLevel<OUTVL_ALL) return;

  bool filter;
  filter = true; //use this to  filter output
  if(filter && spMode){
    ofstream out (debugPxValuesFilename.c_str(), ios::app);
    if (!out){ msgOut(MSG_CRITICAL_ERROR,"Error in opening the file "+
debugPxValuesFilename+" for writing.");}
    int currentYear = MTHREAD->SCD->getYear();
    vector <int> regIds2 = MTHREAD->MD->getRegionIds(2);
    for (uint r=0;r<regIds2.size();r++){
      int rId = regIds2[r];
      //if(rId != 11061) continue;
      ModelRegion* REG = MTHREAD->MD->getRegion(rId);
      vector<Pixel*> regPx = REG->getMyPixels();
      for (uint p=0;p<regPx.size();p++){
        Pixel* px = regPx[p];
        int pxID = px->getID();
        int pxX = px->getX();
        int pxY = px->getY();
        string common = scenarioName + d + i2s(currentYear) + d +
i2s(rId) + d+ i2s(pxID) +d +i2s(pxX)+d+i2s(pxY)+d;

        for(uint f=0;f<fTypes.size();f++){
          double tp_m = px->getMultiplier("tp_multiplier",fTypes[f]);
          common += d2s(tp_m)+d;
        }
        for(uint f=0;f<fTypes.size();f++){
          double m_m = px->getMultiplier("mortCoef_multiplier",
fTypes[f]);
          common += d2s(m_m)+d;
        }

        // First vars by only ft...
        // expectedReturns
        out << common << "expectedReturns" << d;
        for(uint f=0;f<fTypes.size();f++){
          for(uint u=0;u<dClasses.size()-1;u++){
            out << d;
          }
          out << px->expectedReturns[f] << d;
          //out << 0.0 << d;
        }
        out << "\n";
        //----
        out << common <<"vol" << d;
        for(uint f=0;f<fTypes.size();f++){
          for(uint u=0;u<dClasses.size()-1;u++){
            out << px->vol[f][u]<< d;
          }
          out << vSum(px->vol[f]) << d;
        }
        out << "\n";
        //----
        out << common <<"area" << d;
        for(uint f=0;f<fTypes.size();f++){
          for(uint u=0;u<dClasses.size()-1;u++){
            out << px->area[f][u]<< d;
          }
          out << vSum(px->area[f]) << d;
        }
        out << "\n";
        //----
        out << common <<"cumTp_exp" << d;
        for(uint f=0;f<fTypes.size();f++){
          for(uint u=0;u<dClasses.size()-1;u++){
            out << px->cumTp_exp[f][u]<< d;
          }
          out << vSum(px->cumTp_exp[f]) << d;
        }
        out << "\n";
        //----
        out << common <<"vHa_exp" << d;
        for(uint f=0;f<fTypes.size();f++){
          for(uint u=0;u<dClasses.size()-1;u++){
            out << px->vHa_exp[f][u]<< d;
          }
          out << vSum(px->vHa_exp[f]) << d;
        }
        out << "\n";
      } // end for each pixel
    } // end for each region
  } // end filter
} // end function printDebugPixelValues
```

```


/**
This routine clean the output scenario from previous outputs of the defined scenario.
Other scenarios are untouched. The scenarioName must be in the first row.
@param filename Filename of the output file to clean
@param scenarioName Name of the scenario we are replacing
@param d Field delimiter. It must not be changed in the meantime (between the various scenarios)
*/
void
Output::cleanScenario(string fileName, string scenarioName, char d){
  string dStr(&d,1);
  vector <string> rows;
  string tempRow;
  ifstream inFile (fileName.c_str(), ios::in);
  if (!inFile){
    msgOut(MSG_ERROR,"Error in opening the file "+fileName+" for reading.");
    return;
  }
  while( getline (inFile,tempRow) ){
    vector<string> tokens;
    tokenize(tempRow,tokens,dStr);
    if(tokens[0] != scenarioName)
      rows.push_back( tempRow );
  }
  inFile.close();
  ofstream out(fileName.c_str(), ios::out);
  for(uint i=0;i<rows.size();i++){
    out << rows[i];
    out << "\n";
  }
}

```

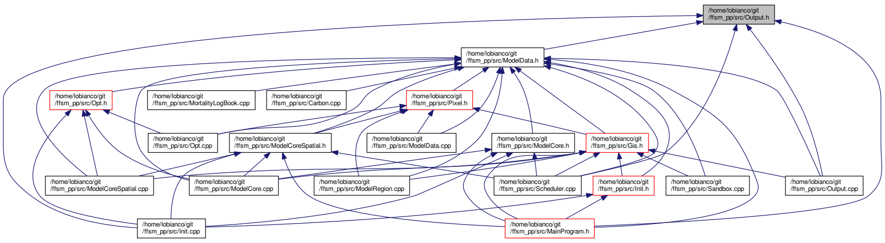
5.115 /home/lobianco/git/ffsm_pp/src/Output.h File Reference

```
#include <string>
#include <vector>
#include <map>
#include <stdexcept>
#include <iostream>
#include <sstream>
#include <time.h>
#include "BaseClass.h"
```

Include dependency graph for Output.h:



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


/home/lobianco/git /ffsm_pp/src/Output.h
/home/lobianco/git /ffsm_pp/src/ModelData.h
/home/lobianco/git /ffsm_pp/src/Opt.h
/home/lobianco/git /ffsm_pp/src/MortalityLogBook.cpp
/home/lobianco/git /ffsm_pp/src/Carbon.cpp
/home/lobianco/git /ffsm_pp/src/Pixel.h
/home/lobianco/git /ffsm_pp/src/Opt.cpp
/home/lobianco/git /ffsm_pp/src/ModelCoreSpatial.h
/home/lobianco/git /ffsm_pp/src/ModelData.cpp
/home/lobianco/git /ffsm_pp/src/ModelCore.h
/home/lobianco/git /ffsm_pp/src/Gis.h
/home/lobianco/git /ffsm_pp/src/ModelCoreSpatial.cpp
/home/lobianco/git /ffsm_pp/src/ModelCore.cpp
/home/lobianco/git /ffsm_pp/src/ModelRegion.cpp
/home/lobianco/git /ffsm_pp/src/Scheduler.cpp
/home/lobianco/git /ffsm_pp/src/Init.h
/home/lobianco/git /ffsm_pp/src/Sandbox.cpp
/home/lobianco/git /ffsm_pp/src/Output.cpp
/home/lobianco/git /ffsm_pp/src/Init.cpp
/home/lobianco/git /ffsm_pp/src/MainProgram.h

Classes

- class Output

    Output methods

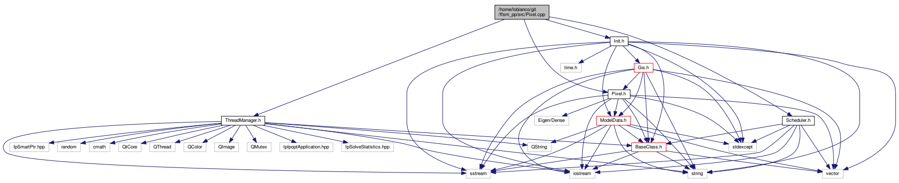
5.116 Output.h

```
/***************************************************************************
 *   Copyright  (C) 2015 by Laboratoire d’Economie Forestière              *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify *
 *   it under the terms of the GNU General Public License as published by *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#ifndef STDOUTPUT_H
#define STDOUTPUT_H

// Core C++ headers
#include <string>
#include <vector>
#include <map>
#include <stdexcept>
#include <iostream>
#include <sstream>
#include <time.h>

//regmas headers
#include "BaseClass.h"


class Pixel;


/// %Output methods

/**
Class responsable to output  the data, both as all kind of log as well as georeferenciated one.
@author Antonello Lobianco
*/
class Output: public BaseClass{
public:
                      Output(ThreadManager* MTHREAD_h); ///< Constructor
                     ~Output();

  void                initOutput();
    void                commonInit();
    void                initOutputMaps();
    void                initOutputForestData();
    void                initOutputProductData();
    void                initOptimisationLog();
    void                initDebugOutput();
    void                initDebugPixelValues();
    void                initCarbonBalance();
  void                print();
    void                printMaps();
    void                printForestData(bool finalFlush=false);
    void                printProductData(bool finalFlush=false);
    void                printCarbonBalance();
  void                printFinalOutput();
  void                printDebugOutput();
  void                printDebugPixelValues();
  void                printOptLog(bool optimal, int &nIterations, double &obj);
  char                getOutputFieldDelimiter();
  void                cleanScenario(string fileName, string scenarioName, char d);

    vector <vector < vector <vector <vector <double> > > > > expReturnsDebug; ///< l2_region, for type, d.c., pr prod, variable name
    vector <string>       expReturnsDebugVariables;

private:
  int                                      oLevel;
  char                                          d;
  int                                      inYear;
  int                                      nYears;
  string                                  baseDir;
  string                                     oDir;
  string                             scenarioName;
  string                                 oFileExt;
  bool                                 oHRedeable;
  bool                                oSingleFile;
  vector<int>                              oYears; // list of output years for data
  vector<int>                          mapsOYears; // list of output years for maps
  int                                   wRegId_l1;
  int                                   wRegId_l2;
  string                              outFileName;
  vector <string>                 outForVariables;
  vector <string>                outProdVariables;
  int                                outStepRange;
  bool                   forestDiamDetailedOutput;
  vector<string>                             priPr;
  vector<string>                             secPr;
  vector<string>                             allPr;
  vector<int>                             l1regIds;
  vector < vector <int> >                      l2r;
  vector <string>                           fTypes;
  vector <string>                         dClasses; /// includes an empty string for variables without diameter attribute
  vector <string>                        pDClasses; ///< production diameter classes: exclude the fist diameter class below 15 cm
  int                                     nPriPr;
  int                                     nSecPr;
  int                                     nAllPr;
  int                                       nL2r;
  string                             logFilename;
  string                           debugFilename;
  string                   debugPxValuesFilename;
  bool                                    spMode; // spatial mode
};
#endif
```

5.117 /home/lobianco/git/ffsm_pp/src/Pixel.cpp File Reference

```
#include "Pixel.h"
#include "ThreadManager.h"
#include "Scheduler.h"
#include "Init.h"
```

Include dependency graph for Pixel.cpp:



5.118 Pixel.cpp

```cpp
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d'Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#include "Pixel.h"
#include "ThreadManager.h"
#include "Scheduler.h"
#include "Init.h"

Pixel::Pixel(double ID_h, ThreadManager* MTHREAD_h): ID(ID_h)
{
  MTHREAD=MTHREAD_h;
  int nft = MTHREAD->MD->getForTypeIds().size();
  vector<double> temp(nft,1);
  //vector<double> temp2(nft,0);
  spMods = temp;
  avalCoef = 1;
  //vMort  = temp2;
  //std::fill(v.begin(), v.end(), 0);
}

Pixel::~Pixel()
{
}

/**
The function return a vector of pointers to Pixels at the gived distance from the caller pixel.\\
The list start with those on the Top, then add those on the right, those on the bottom and those on the left. Finally it had the corner pixels (that are more far).\\
It takes into consideration borders correctly.

Fully tested on internal points as well semi-border cases, border cases and corner cases. ALL OK.

@param distLevel_h Distance in number of adiacent pixels. It has to be at least 1 (the function return an error if it is 0).
*/
vector <Pixel *>
Pixel::getPixelsAtDistLevel(int distLevel_h) const{

  if (distLevel_h<1) {
    msgOut(MSG_CRITICAL_ERROR, "getPixelsAtDistLevel() is defined for distances of at least 1 !");
  }

  vector <Pixel *> toReturn;
  int xNPixels = MTHREAD->GIS->getXNPixels();
  int yNPixels = MTHREAD->GIS->getYNPixels();
  int thisX = this->getX();
  int thisY = this->getY();
  int minX = max(0        , (thisX - distLevel_h)+1);
  int maxX = min(xNPixels , thisX + distLevel_h);
  int minY = max(0        , (thisY - distLevel_h)+1);
  int maxY = min(yNPixels , thisY + distLevel_h);

  // getting the top pixels (corner exluded)...
  if (thisY-distLevel_h >=0){
    for(int i=minX;i<maxX;i++){
      toReturn.push_back(MTHREAD->GIS->getPixel(i,thisY-distLevel_h));
    }
  }
  // getting the right pixels (corner exluded)...
  if (thisX+distLevel_h < xNPixels){
    for(int i=minY;i<maxY;i++){
      toReturn.push_back(MTHREAD->GIS->getPixel(thisX+distLevel_h,i));
    }
  }
  // getting the bottom pixels (corner exluded)...
  if (thisY+distLevel_h < yNPixels){
    for(int i=minX;i<maxX;i++){
      toReturn.push_back(MTHREAD->GIS->getPixel(i,thisY+distLevel_h));
    }
  }
  // getting the left pixels (corner exluded)...
  if (thisX-distLevel_h >= 0){
    for(int i=minY;i<maxY;i++){
      toReturn.push_back(MTHREAD->GIS->getPixel(thisX-distLevel_h,i));
    }
  }

  // getting the corners (correctly at the end, after already retrieved the other pixels..)...
  // top-left..
  if (thisX-distLevel_h >= 0 && thisY-distLevel_h >=0){
    toReturn.push_back(MTHREAD->GIS->getPixel(thisX-distLevel_h,thisY-distLevel_h));
  }
  // top-right..
  if (thisX+distLevel_h < xNPixels && thisY-distLevel_h >=0){
    toReturn.push_back(MTHREAD->GIS->getPixel(thisX+distLevel_h,thisY-distLevel_h));
  }
  // bottom-right..
  if (thisX+distLevel_h < xNPixels && thisY+distLevel_h <yNPixels){ // bug discovered 20070719
    toReturn.push_back(MTHREAD->GIS->getPixel(thisX+distLevel_h,thisY+distLevel_h));
  }
  // bottom-left..
  if (thisX-distLevel_h >= 0 && thisY+distLevel_h <yNPixels){
    toReturn.push_back(MTHREAD->GIS->getPixel(thisX-distLevel_h,thisY+distLevel_h));
  }
  return toReturn;
}

/*void //moved as inline function
Pixel::setValue(const string & layerName_h, const double & value_h){

  //tempValuePair.first = layerName_h;     // type of first is string
  //tempValuePair.second = value_h;            // type of second is double
  //tempValuePair = make_pair (layerName_h,value_h);
  values.insert(pair<string, double>(layerName_h, value_h));
  //values.insert(tempValuePair);


}*/

/*
inline void
Pixel::setValue (const string& parName, const string& forName, const string& dClass, const int& year, const double& value_h){
  values.insert(pair<string, double>(MTHREAD->GIS->pack(parName, forName, dClass, year), value_h));
}
*/


void
Pixel::changeValue(const string &layerName_h, const double &value_h, const bool & setNoValueForZero){
  map<string, double>::iterator p;
  p=values.find(layerName_h);
  if(p != values.end()){
    if(setNoValueForZero && value_h == 0){
      p->second = MTHREAD->GIS->getNoValue();
    } else {
      p->second = value_h;
    }
  } else {
      msgOut(MSG_ERROR, "Coud not change pixel value for layer "+layerName_h+". Layer don’t found.");
  }
  return;
```

```
}

/*
void
Pixel::changeValue (const double &value_h, const string& parName, const string& forName, const string &dClass, const int &year, const bool &setNoValueForZero){
  changeValue(MTHREAD->GIS->pack(parName, forName, dClass, year), value_h, setNoValueForZero);
}
*/

double
Pixel::getDoubleValue(const string &layerName_h, const bool &returnZeroForNoValue) const{
  vIter=values.find(layerName_h);
  if(vIter != values.end()) {
    if(returnZeroForNoValue && vIter->second==MTHREAD->GIS->getNoValue()){
      return 0.0;
    } else {
      return vIter->second;
    }
  } else {
    msgOut(MSG_WARNING, "No layer \""+layerName_h+"\" found on pixel ("+i2s(getX())+","+i2s(getY())+"). Sure you didn't mispelled it?");
    if(returnZeroForNoValue){
      return 0.0;
    } else {
      return MTHREAD->GIS->getNoValue();
    }
  }
}

/**
getMultiplier() returns the value of the multiplier as memorized in the spatialDataSubfolder layers or in the forData table.
It will looks for exact match or for lower years if available.
If no layers are available or the usePixelData option is not used, it will return 1.
If the tp_multiplier is asked for, it will adjusts for spatial variance coefficient.
If the mortCoef_multiplier is used and we are in the table settings it will adjust it by mortCoef_link.
*/
double
Pixel::getMultiplier (const string& multiplierName, const string& forName, int year){


  if(year==DATA_NOW){year = MTHREAD->SCD->getYear();}


    double multiplierSpVar = (multiplierName == "tp_multiplier")?getSpModifier(forName):1.0;

    vector <string> modifiersFromTable = MTHREAD->MD->getStringVectorSetting("modifiersFromTable");

    if(std::find(modifiersFromTable.begin(), modifiersFromTable.end(), multiplierName) != modifiersFromTable.end()) {
      // load multiplier from forData table..
      int regId = getMyRegion()->getRegId();
      double multiplier = MTHREAD->MD->getForData(multiplierName, regId, forName, "", year);
      if (multiplierName == "mortCoef_multiplier"){
        return pow(multiplier,MTHREAD->MD->getDoubleSetting("mortMultiplier_link"))*multiplierSpVar; //Added to account that our multipliers are based on probability of presence and not on planted/managed forests, where mortality is somhow reduced
      }
      return multiplier*multiplierSpVar;

    } else {
      // load multiplier from layer file..

      // return 1 if not using pixel mode
      if(!MTHREAD->MD->getBoolSetting("usePixelData")) return 1.0;
      string search_for = multiplierName+"#"+forName+"##"+i2s(year);
      map <string,double>::const_iterator i = values.upper_bound(search_for); //return the position always upper to the found one, even if it's an equal match.
      if(i!= values.begin())  i--; // this rewind the position to the one just before or equal
      const string& key = i->first;
      string search_base = search_for.substr(0,search_for.size()-4);
      if (key.compare(0, search_base.size(), search_base) == 0){
          //cout << "MATCH: " << search_for <<", "<< i->first << ", " << i->second << endl;
          //if(i->second != 1){
          //  cout << "NOT ONE: " << search_for <<", "<< i->first << ", " << i->second << endl;
          //  exit(0);
          //}
          return i->second*multiplierSpVar;
      } else {
          //cout << "NOTM:  " << search_for <<", "<< i->first << endl;
          return 1.0*multiplierSpVar;
      }
```

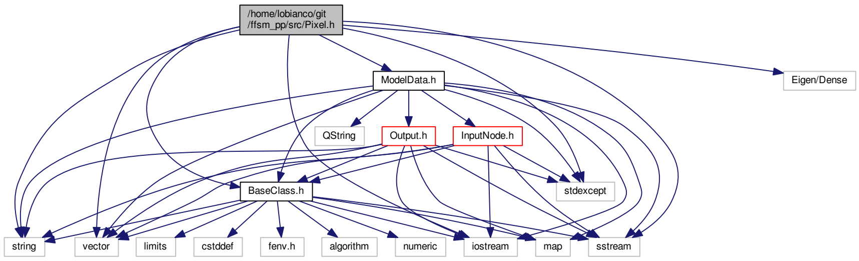
```cpp

  }
}

/**
The mortality returned is the increased yearly mortality due to any affecting pathogenes.
The function load the relevant pathogen mortality rule(s), for each of them check for how many years the phatogen is present with concentrations
above the threshold and returns the relavant increase in mortality (summing them in case of multiple pathogens).

*/
double
Pixel::getPathMortality(const string& forType, const string& dC, int year){
  if(!MTHREAD->MD->getBoolSetting("usePathogenModule")) return 0.0;

  string debug=forType;
  int initialOptYear = MTHREAD->MD->getIntSetting("initialOptYear");
  int simulationYears = MTHREAD->MD->getIntSetting("simulationYears");

  int maxYear = initialOptYear + simulationYears;

  vector<pathRule*> pathRules = MTHREAD->MD->getPathMortalityRule(forType,dC);

  double pathMort = 0.0;
  if(year==DATA_NOW){year = MTHREAD->SCD->getYear();}

  for(uint r=0;r<pathRules.size();r++){
    string pathId=pathRules[r]->pathId;
    double pres_min=pathRules[r]->pres_min;
    vector<double> mortCoefficients=pathRules[r]->mortCoefficents;
    double pathMort_thispath = 0.0;
    for(uint y=year;y>(year-mortCoefficients.size());y--){
      int i =year-y;
      int y2 = y;
      if(y>=maxYear){
        y2=maxYear-1;
      }

      string layerName="pathogen_pp#"+pathId+"#"+i2s(y2);
      if(MTHREAD->GIS->layerExist(layerName)){
        if (this->getDoubleValue(layerName,true)>= pres_min){
          pathMort_thispath = mortCoefficients[i];
        }
      }
    }
    pathMort += pathMort_thispath;
  }
  return pathMort;

}

void
Pixel::correctInputMultiplier (const string& multiplierName, const string& forName, double coefficient){
  string search_for = multiplierName+"#"+forName+"#";
  for (std::map<string,double>::iterator it=values.lower_bound(search_for); it!=values.end(); ++it){
    if (it->first.compare(0, search_for.size(), search_for) == 0){
          //cout << ID << ";" << forName << ";" << coefficient << endl;
      it->second = it->second * coefficient;
    }
  }
}

double
Pixel::getDoubleValue (const string& parName, const string& forName, const string& dClass, const int& year, const  bool& returnZeroForNoValue){
  return getDoubleValue(MTHREAD->GIS->pack(parName, forName, dClass, year), returnZeroForNoValue);
}

void
Pixel::newYear(){

}

double
Pixel::getPastRegArea(const int& ft_idx, const int& year){
  map <int,vector<double> >::const_iterator i=regArea.find(year);
  if(i != regArea.end()) {
    return i->second.at(ft_idx);
  } else {
    msgOut(MSG_ERROR, "Asking for a pastRegArea of a not-registered year. I don’t have year "+i2s(year)+"!");
  }
}
```

```

void
Pixel::setPastRegArea(const double& value, const int& ft_idx, const int& year){
  msgOut(MSG_CRITICAL_ERROR,"TODO");
  /*map <int,vector<double> >::const_iterator i=regArea.find(year);
  if(i != regArea.end()) {
    // we already have this year, let’s see if the vector is bif enough
    int currsize = i->second.size();
    for(j=0;j<ft_idx-currside;j++){

    }
    return i->second.at(ft_idx);
  } else {
    // new year
  }


  pair<int,vector<double> newRegArea;
  */


}

void
Pixel::swap(const int& swap_what){
  switch (swap_what){
    case VAR_VOL:
      vol_l = vol;
      break;
    case VAR_AREA:
      area_l = area;
      break;
    default:
      msgOut(MSG_CRITICAL_ERROR,"Don’t know how to swap "+swap_what);
  }
}


double
Pixel::getSpModifier(const string& ft){
  vector<string>ftypes = MTHREAD->MD->getForTypeIds();
  for (int i=0;i<ftypes.size();i++){
      if (ftypes[i] == ft){
          return spMods.at(i);
      }
  }
  msgOut(MSG_CRITICAL_ERROR,"Asked spatial modifier for a forest type that doesn’t exist");

}

ModelRegion*
Pixel::getMyRegion(const int& rLevel){
    if(rLevel==2){
        return l2region;
    } else if (rLevel==1) {
        return l2region->getParent();
    } else {
        msgOut(MSG_ERROR, "Requested a unknown level region code in getMyRegion().");
    }
}
```

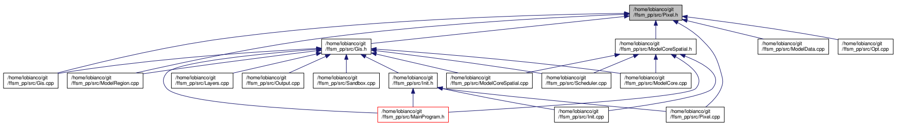
5.119 /home/lobianco/git/ffsm_pp/src/Pixel.h File Reference

```
#include <string>
#include <vector>
#include <stdexcept>
#include <iostream>
#include <sstream>
#include <Eigen/Dense>
#include "BaseClass.h"
#include "ModelData.h"
```

Include dependency graph for Pixel.h:



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



Classes

- class Pixel

  Pixel-level class.

5.120 Pixel.h

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d'Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#ifndef PIXELS_H
#define PIXELS_H

#include <string>
#include <vector>
#include <stdexcept>
#include <iostream>
#include <sstream>

#include <Eigen/Dense>

// regmas headers...
#include "BaseClass.h"
#include "ModelData.h"

```

```
class Gis;        //forward declaration

using namespace Eigen;

/// Pixel-level class

/**
This class manage the info at the pixel level. A vector of pixel objects is owned by the class Gis.
@author Antonello Lobianco
*/
class Pixel: public BaseClass{

public:
                      Pixel(double ID_h, ThreadManager* MTHREAD_h);
                     ~Pixel();

  /// Return the value for a specific layer
  double              getDoubleValue (const string& layerName_h, const  bool& returnZeroForNoValue = false) const;
  double              getDoubleValue (const string& parName, const string& forName, const string& dClass, const int& year, const  bool& returnZeroForNoValue = false);
  double              getMultiplier (const string& multiplierName, const string& forName, int year=DATA_NOW);
  double              getPathMortality(const string& forType, const string& dC, int year=DATA_NOW);                ///< Return the INCREASED mortality due to pathogen presence for a given ft and dc in a certain year (default the running year)
  void                correctInputMultiplier (const string& multiplierName, const string& forName, double coefficient=1); ///< It apply a given coefficient to all the multipliers layers of a given ft
  void                newYear();
  double              getPastRegArea(const int& ft_idx, const int& year);
  void                setPastRegArea(const double& value, const int& ft_idx, const int& year);
  ModelRegion*        getMyRegion(const int& rLevel = 2);


  // space..
  double              getID() const {return ID;}  ;
  int                 getX() const {return pxX;} ;
  int                 getY() const {return pxY;};
  /// Return a vector of pixels at the specified distance (in levels, not in physical units)
  vector <Pixel *>    getPixelsAtDistLevel (int distLevel_h) const;

  string              getPxComments() const {return pxComments;};
  double              getCachedDouble() const {return cachedDouble;};


  /// Insert a new layer and its value
  void                setValue ( const string& layerName_h, const double& value_h ){values.insert(pair<string, double>(layerName_h, value_h));}
  //inline void         setValue ( const string& parName, const string& forName, const string& dClass, const int& year, const double& value_h); // never used ???
  /// Change the value of an existing layerMTHREAD->GIS->pack(parName, forName, dClass, year), value_h,
  void                changeValue (const string& layerName_h, const double& value_h, const bool& setNoValueForZero=false );
  //void                changeValue (const double& value_h, const string& parName, const string& forName, const string& dClass, const int& year, const bool& setNoValueForZero=false);
  void                setCoordinates ( int x_h, int y_h ) {pxX=x_h; pxY=y_h;};
  void                setPxComments ( std::string pxComments_h ) {pxComments = pxComments_h;};
  void                setCachedDouble(double cachedDouble_h){cachedDouble=cachedDouble_h;};
  void                clearCache(){cachedDouble=0;};
  void                setSpModifier(const double& value, const int& ftindex){spMods.at(ftindex)=value;};
  double              getSpModifier(const string& ft);
  void                swap(const int &swap_what); ///< Assign to the delayed value the current values, e.g. vol_l = vol
  void                setMyRegion(ModelRegion* region_h){l2region = region_h;};

  // matrices of (ft,dc)
  /*MatrixXd                                   vol;
  MatrixXd                                  area;
  MatrixXd                               regArea;
  MatrixXd                                  hVol;
  MatrixXd                                 vol_l;
  MatrixXd                                area_l;
  MatrixXd                             regArea_l;
  MatrixXd                                hVol_l;
  MatrixXd                                  beta;
  MatrixXd                                  mort;
  MatrixXd                                    tp;
  MatrixXd                                 cumTp;*/

  vector <vector <double> >                  vol; // by ft,dc
  vector <vector <double> >                 area; // by ft,dc
  vector <double>                 initialDc0Area; // by ft
  vector <vector <double> >                hArea; // by ft, dc  // possibly in ha, but to be check for 100% sure
  vector <vector <double> >                 hVol; // by ft.dc
  vector < vector <vector <double> > >hVol_byPrd; // by ft, dc, pp
```

```
 map <int, vector <double> >            regArea; // by year, ft
 //vector <double>                             in; // by pp
 //vector <double>                             hr; // by pp
 vector <double>                            vReg; // by ft
 vector <vector <double> >                 vMort; // by ft,dc
 vector <double>                 expectedReturns; // by ft
 vector <double>     expectedReturnsNotCorrByRa; ///< by ft. Attenction,
reported expReturns at "forest" level (compared with those at forest type level) do NOT include ra

 vector <vector <double> >                 vol_l; ///< store the volumes of the previous year
 vector <vector <double> >                area_l; ///< store the areas of the previous year

 vector <vector <double> >                  beta;
 vector <vector <double> >                  mort;
 vector <vector <double> >                    tp;
 vector <vector <double> >                 cumTp; ///< This is time of passage to REACH a diameter
class (while the exogenous tp by diameter class is the time of passage to LEAVE to the next d class)
 vector <vector <double> >                   vHa; ///< Volume at hectar by each diameter class [m^3/ha]
 vector <vector <double> >              cumAlive; ///< Cumulative prob of remaining alive at
beginnin of a given diam class
 vector <vector <double> >             cumTp_exp; ///< This is the **expected** version of cumTp,
used for calculating profits
 vector <vector <double> >               vHa_exp; ///< This is the **expected** version of vHa, used
for calculating profits
 vector <vector <double> >          cumAlive_exp; ///< This is the **expected** version of
cumAlive, used for calculating profits

 // management variables (pixel==agent)
 double                          portfolioVarRa; ///< Sampling derived risk aversion on
portfolio variance for of this agent
 double                                 expType; ///< Sampling derived expectation types of this
agent (forest bilogical parameters: growth, mortality)
 double                           expTypePrices; ///< Sampling derived expectation types of
this agent (prices)
 bool                              usePortfolio; ///< Sampling derived usage of portfolio
management (false/true)
 double                                avalCoef; ///< Availability (of wood resources)
coefficient. A [0,1] coefficient that reduce avaiability of wood resources to exploitation due to local reasons
(protected area, altimetry..)

private:
 map<string, double>            values;  ///< Map of values for each layer
 mutable map<string, double>::const_iterator vIter; //< Iterator for the map of values
 double                             ID;
 int                               pxX;
 int                               pxY;
 string                     pxComments;
 double                   cachedDouble;  ///< Cachable double used in some optimized
algorithms
 vector<double>                 spMods;  ///< The sampled spatial modifiers (by forest type)
 ModelRegion*                 l2region;  ///< Pointer to level 2 region where this
pixel is

};

#endif
```

5.121 /home/lobianco/git/ffsm_pp/src/resources.qrc File Reference

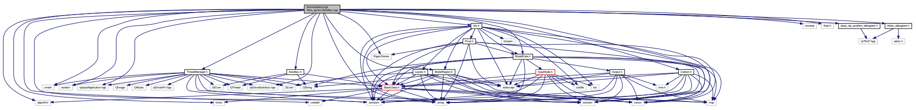
5.122 resources.qrc

```
<RCC>
    <qresource prefix="/" >
        <file>imgs/clear.png</file>
        <file>imgs/exit.png</file>
        <file>imgs/help.png</file>
        <file>imgs/icon.png</file>
        <file>imgs/info.png</file>
        <file>imgs/open.png</file>
        <file>imgs/options.png</file>
        <file>imgs/pause.png</file>
        <file>imgs/play.png</file>
        <file>imgs/save.png</file>
        <file>imgs/saveas.png</file>
        <file>imgs/showHideLogArea.png</file>
        <file>imgs/stop.png</file>
        <file>imgs/view-refresh.png</file>
    </qresource>
</RCC>
```

5.123 /home/lobianco/git/ffsm_pp/src/Sandbox.cpp File Reference

```
#include <algorithm>
#include <cmath>
#include <map>
#include <Eigen/Dense>
#include "Sandbox.h"
#include "ThreadManager.h"
#include "ModelData.h"
#include "Gis.h"
#include "ModelRegion.h"
#include "Carbon.h"
#include <iostream>
#include <iomanip>
#include <string>
#include <random>
#include <float.h>
#include <limits>
#include <cstddef>
#include "IpIpoptApplication.hpp"
#include "IpSolveStatistics.hpp"
#include "Ipopt_nlp_problem_debugtest.h"
#include "Adolc_debugtest.h"
```

Include dependency graph for Sandbox.cpp:



Classes

- struct GccTest

Typedefs

- typedef map< string, string > TStrStrMap
- typedef pair< string, string > TStrStrPair

Functions

- template<class T >
  vector< T > getVectorSetting (string name_h, int type)

5.123.1 Typedef Documentation

5.123.1.1 typedef map<string, string> TStrStrMap

Definition at line 75 of file Sandbox.cpp.

5.123.1.2 typedef pair< string, string > TStrStrPair

Definition at line 76 of file Sandbox.cpp.

5.123.2 Function Documentation

5.123.2.1 vector< T > getVectorSetting (string *name_h*, int *type*)

Definition at line 1291 of file Sandbox.cpp.

```
01291                                                                            {
01292
01293     vector <string> myStringDatas;
01294     myStringDatas.push_back("aaaaa");
01295     myStringDatas.push_back("bbbbb");
01296     myStringDatas.push_back("ccccc");
01297     vector <T> xVector;
01298
01299     for (int i=0;i<myStringDatas.size();i++){
01300       istringstream iss(myStringDatas[i]);
01301       T x;
01302       iss >> x;
01303       xVector.push_back(x);
01304     }
01305
01306     return xVector;
01307 }
```

5.124 Sandbox.cpp

```
00001 /***************************************************************************
00002  *   Copyright (C) 2015 by Laboratoire d'Economie Forestière               *
00003  *   http://ffsm-project.org                                               *
00004  *                                                                         *
00005  *   This program is free software; you can redistribute it and/or modify  *
00006  *   it under the terms of the GNU General Public License as published by  *
00007  *   the Free Software Foundation; either version 3 of the License, or     *
00008  *   (at your option) any later version, given the compliance with the     *
00009  *   exceptions listed in the file COPYING that is distribued together     *
00010  *   with this file.                                                       *
00011  *                                                                         *
00012  *   This program is distributed in the hope that it will be useful,       *
00013  *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
00014  *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
00015  *   GNU General Public License for more details.                          *
00016  *                                                                         *
00017  *   You should have received a copy of the GNU General Public License     *
00018  *   along with this program; if not, write to the                         *
00019  *   Free Software Foundation, Inc.,                                       *
00020  *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
00021  ***************************************************************************/
00022 #include <algorithm>
00023
00024 #include <cmath>
00025 #include <algorithm>
00026 #include <map>
00027
00028 #include <Eigen/Dense>
00029
00030 #include "Sandbox.h"
00031 #include "ThreadManager.h"
00032 #include "ModelData.h"
00033 #include "Gis.h"
00034 #include "ModelRegion.h"
00035 #include "Carbon.h"
00036
00037
00038 //Testing random distribution, using some new in C++ random generator seeds..
00039 #include <iostream>
00040 #include <iomanip>
00041 #include <string>
00042 #include <map>
00043 #include <random>
```

```
#include <cmath>
#include <float.h>
#include <limits>
#include <cstddef>


// Testing zip library...
//#include "zip.h"
//#include "unzip.h"
//#include <QFile>
//#include <QFileInfo>
//#include <QString>
//#include <QStringList>
//#include <QList>
//#include <iostream>
//#include <iomanip>


//Testing FlopC++ (requires modified src.pro qmake file)
//#include "flopc.hpp"
//using namespace flopc;
//#include <OsiClpSolverInterface.hpp>
//#include <OsiCbcSolverInterface.hpp>

#include "IpIpoptApplication.hpp"
#include "IpSolveStatistics.hpp"

#include "Ipopt_nlp_problem_debugtest.h"
#include "Adolc_debugtest.h"


typedef map<string, string> TStrStrMap;
typedef pair<string, string> TStrStrPair;

using namespace std;

Sandbox::Sandbox(ThreadManager* MTHREAD_h){
  MTHREAD=MTHREAD_h;
}

Sandbox::Sandbox(){

}


Sandbox::~Sandbox(){

}

// ---------------------------------------------
struct GccTest
{

  GccTest(string name_h){
    nameMember = name_h;
  };

  string nameMember;

   operator string ()
   {

    cout << "the first function\n";
    cout << nameMember << endl;
    return "42";
   }

   operator int ()
   {
    cout << "its \"underload\"\n";
      return 42;
   }

  operator vector<int> ()
  {
    cout << "within vector <int>" << endl;
    vector <int> toReturn;
    toReturn.push_back(3);
    toReturn.push_back(4);
    toReturn.push_back(5);
    return toReturn;
  }

};

// --------------------------------------
void
```

```
Sandbox::basicTest(){

  /*
  // Testing debugging a map
  iisskey k1(2007,11021,"broadL_HighF","15");
  iisskey k2(2007,11021,"broadL_HighF","30");
  iisskey k3(2007,11021,"con_HighF","15");
  iisskey k4(2007,11022,"broadL_HighF","15");
  iisskey k5(2008,11021,"broadL_HighF","15");

  // Testing the new changeMapValue(), incrMapValue(), resetMapValues(), incrOrAddMapValue(map, key, value) and vectorToMap() funcions
  map<iisskey,double> testMap;
  pair<iisskey,double> pair1(k1,1.1);
  pair<iisskey,double> pair2(k2,1.2);
  pair<iisskey,double> pair3(k3,1.3);
  pair<iisskey,double> pair4(k4,1.4);
  pair<iisskey,double> pair5(k5,1.5);
  testMap.insert(pair1);
  testMap.insert(pair2);
  testMap.insert(pair3);
  testMap.insert(pair4);
  testMap.insert(pair5);
  debugMap(testMap,iisskey(NULL,NULL,"",""));
  debugMap(testMap,iisskey(2007,NULL,"con_HighF",""));
  exit(0);
  */




 /*
 // Testing standard deviation algorithm, as from http://stackoverflow.com/questions/7616511/calculate-mean-and-standard-deviation-from-a-vector-of-samples-in-c-using-boos
 vector<double> v;
 v.push_back(3.0);
 v.push_back(2.0);
 v.push_back(5.0);
 v.push_back(4.0);
 double sum = std::accumulate(std::begin(v), std::end(v), 0.0);
 double m =  sum / v.size();
 double accum = 0.0;
 std::for_each (std::begin(v), std::end(v), [&](const double d) {
   accum += (d - m) * (d - m);
 });
 double stdev = sqrt(accum / (v.size()-1));
 cout << stdev << endl;
 double sd2 = getSd(v);
 double sd3 = getSd(v,false);
 cout << sd2 << endl;
 cout << sd3 << endl;
 exit(0);
 */

 /*
 // Testing tokenize, untokenize functions
 vector<string> istrings;
 istrings.push_back("Questo");
 istrings.push_back("cielo");
 istrings.push_back("è");
 istrings.push_back("sempre");
 istrings.push_back("più");
 istrings.push_back("blu.");
 string delimiter = " . ";

 string fullstring="";
 vector<string> ostrings;
 untokenize(fullstring, istrings, delimiter);
 cout << fullstring << endl;

 fullstring += delimiter;
 cout << fullstring << endl;

 tokenize(fullstring, ostrings, delimiter);
 for (uint i=0;i<ostrings.size();i++){
  cout << ostrings[i] << endl;
 }
 exit(0);
 */


 /*
 // Testing FlopC++
 // For a single file compile as:
 // -- two passages:
 // g++ -O3 -I /usr/include/coin -DFLOPCPP_BUILD `PKG_CONFIG_PATH=/usr/lib64/pkgconfig:/usr/lib/pkgconfig:/usr/share/pkgconfig: pkg-config --libs flopcpp osi-cbc osi-clp` transport.cpp -c -o  transport.o
```

```
  // g++ -o transport2 transport.o -Wl,-rpath,'$ORIGIN' -L . -DFLOPCPP_BUILD `PKG_CONFIG_PATH=/usr/lib64/
pkgconfig:/usr/lib/pkgconfig:/usr/share/pkgconfig: pkg-config --libs flopcpp osi-cbc osi-clp`
  // -- single passage:
  // g++ -O3 -I /usr/include/coin transport.cpp -DFLOPCPP_BUILD `PKG_CONFIG_PATH=/usr/lib64/pkgconfig:/usr/
lib/pkgconfig:/usr/share/pkgconfig: pkg-config --libs flopcpp osi-cbc osi-clp`  -o  transport3

  MP_model::getDefaultModel().setSolver(new OsiClpSolverInterface);
  //MP_model::getDefaultModel().setSolver(new OsiCbcSolverInterface);
  enum  {seattle, sandiego, numS};
  enum  {newyork, chicago, topeka,numD};

  MP_set S(numS);          // Sources
  MP_set D(numD);          // Destinations
  MP_subset<2> Link(S,D);  // Transportation links (sparse subset of S*D)

  Link.insert(seattle,newyork);
  Link.insert(seattle,chicago);
  Link.insert(sandiego,chicago);
  Link.insert(sandiego,topeka);

  MP_data SUPPLY(S);
  MP_data DEMAND(D);

  SUPPLY(seattle)=350;  SUPPLY(sandiego)=600;
  DEMAND(newyork)=325;  DEMAND(chicago)=300;  DEMAND(topeka)=275;

  MP_data COST(Link);

  COST(Link(seattle,newyork)) = 2.5;
  COST(Link(seattle,chicago)) = 1.7;
  COST(Link(sandiego,chicago))= 1.8;
  COST(Link(sandiego,topeka)) = 1.4;

  COST(Link) = 90 * COST(Link) / 1000.0;

  MP_variable x(Link);
  x.display("...");

  MP_constraint supply(S);
  MP_constraint demand(D);

  supply.display("...");

  supply(S) =  sum( Link(S,D), x(Link) ) <= SUPPLY(S);
  demand(D) =  sum( Link(S,D), x(Link) ) >= DEMAND(D);

  cout<<"Here"<<endl;

  minimize( sum(Link, COST(Link)*x(Link)) );
  assert(MP_model::getDefaultModel()->getNumRows()==5);
  assert(MP_model::getDefaultModel()->getNumCols()==4);
  assert(MP_model::getDefaultModel()->getNumElements()==8);
  assert(MP_model::getDefaultModel()->getObjValue()>=156.14 &&
 MP_model::getDefaultModel()->getObjValue()<=156.16);

  x.display("Optimal solution:");
  supply.display("Supply dual solution");
  cout<<"Test transport passed."<<endl;
  */



  /*
  // Testing limits for 0
  double test = DBL_MIN;
  cout << test << endl;
  test = numeric_limits<double>::min();
  cout << test << endl;
  exit(0);
  */


  /*
  // Testing getMaxPos()
  vector<double> test {7,2,6,4,7,2,5,7,2};
  double maxpos   = getMaxPos(test);
  double maxvalue = getMax(test);
  double minpos   = getMinPos(test);
  double minvalue = getMin(test);
  //double maxpos = testB();
  cout << "maxpos: " << maxpos << endl;
  cout << "maxvalue: " << maxvalue << endl;
  cout << "minpos: " << minpos << endl;
  cout << "minvalue: " << minvalue << endl;
  exit(0);
  */


  /*
  //This was in ModelData::debug():
  // ********** START DEBUG CODE....... ************
  double ddebuga=0; //20080209
  uint idebuga=0;
  double ddebugb=0; //20080209
  uint idebugb=0;
  double ddebugc=0; //20080209
  uint idebugc=0;
  double debugmin = 0;
  double debugmax = 1000;
  for (uint q=0;q<10000;q++){
    ddebuga += debugmin + ( (double)rand() / ((double)(RAND_MAX)+(double)(1)) )*(debugmax-debugmin+1);
    ddebugb += debugmin + ( (double)rand() / ((double)(RAND_MAX)+(double)(1)) )*(debugmax-debugmin+1);
    ddebugc += debugmin + ( (double)rand() / ((double)(RAND_MAX)+(double)(1)) )*(debugmax-debugmin+1);
  }
  idebuga = ddebuga;
  idebugb = ddebugb;
  idebugc = ddebugc;
  cout << "idebuga: "<<idebuga<<endl;
  cout << "idebugb: "<<idebugb<<endl;
  cout << "idebugc: "<<idebugc<<endl;
  throw 2;
  // ******** .....END DEBUG CODE *******************
  */

    /*
    // Testing the new iskey class
    iskey op1(2100,"test");
    iskey op2(2100,"test");
    iskey op3(2101,"test");
    iskey op4(2101,"tgst");
    iskey op5(2101,"tb");
    iskey op6(2101,"testa");
    if(op1 == op2){
      cout << "op1 and op2 are equal" << endl;
    }
    if(op1 == op3){
      cout << "op1 and op3 are equal" << endl;
    }
    if(op6 > op3) cout << "test3 passed" << endl;
    if(op5 < op3) cout << "test4 passed" << endl;
    if(op6 >= op3) cout << "test5 passed" << endl;
    if(op6 != op3) cout << "test6 passed" << endl;
    if(op4 <= op3) cout << "test7 passed that it shoudn't" << endl;
    exit(0);
    */

    /*
    // Testing the new changeMapValue(), incrMapValue(), resetMapValues(), incrOrAddMapValue(map, key, value) and vectorToMap() funcions
    map<int,double> testMap;
    for (uint i=0;i<5;i++){
        pair<int,double> mypair(i,i*2.5);
        testMap.insert(mypair);
    }
    double result = findMap(testMap,3,MSG_NO_MSG);
    double result2 = findMap(testMap,1,MSG_ERROR);
    double result3 = findMap(testMap,7,MSG_DEBUG);
    cout << findMap(testMap,3,MSG_NO_MSG)<< endl;
    changeMapValue(testMap,3,200.0,MSG_ERROR);
    cout << findMap(testMap,3,MSG_NO_MSG)<< endl;
    incrMapValue(testMap,3,5.0,MSG_ERROR);
    cout << findMap(testMap,3,MSG_NO_MSG)<< endl;
    incrOrAddMapValue(testMap, 3, 200.0);
    cout << findMap(testMap,3,MSG_NO_MSG)<< endl;
    incrOrAddMapValue(testMap, 10, 100.0);
    cout << findMap(testMap,10,MSG_NO_MSG)<< endl;
    cout << "done" << endl;

    vector<string> mykeys;
    mykeys.push_back("andrea");
    mykeys.push_back("antonello");
    mykeys.push_back("paolo");
    map<string,double> mymap = vectorToMap(mykeys,15.0);
    string searchkey;
    searchkey = "andrea";
    cout << findMap(mymap,searchkey,MSG_DEBUG)<< endl;
    resetMapValues(mymap,32.0);
    cout << findMap(mymap,searchkey,MSG_DEBUG)<< endl;
    exit(0);
    */



    /*
    // ------------------------------------------------------------------
    // Sampling from uniform distribution with local random seed
    // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

    //this code sample from a uniform distribution. In this case also the seed is reinitialisated, but it it valid only locally: the rest of the program run with the same seed

    std::random_device rd;
    std::mt19937 gen(rd());
    std::uniform_int_distribution<> dis(1, 6);

    for (int n=0; n<10; ++n)
        std::cout << dis(gen) << ' ';
    std::cout << '\n';
    exit(0);
   */



  /*
  // ------------------------------------------------------------------
  // Testing how to get all elements in a map by substrings
  // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  map <string,double> values;
  pair <string,double> val1("AAAAAA",1);
  pair <string,double> val2("AAABBB",2);
  pair <string,double> val3("BBBAAA",3);
  pair <string,double> val4("BBBBBB",4);
  pair <string,double> val5("CCCAAA",5);
  pair <string,double> val6("C",6);
  pair <string,double> val7("BBB",7);

  values.insert(val1);
  values.insert(val2);
  values.insert(val3);
  values.insert(val4);
  values.insert(val5);
  values.insert(val6);
  values.insert(val7);

  cout << "Printing whole map" << endl;
  for (std::map<string,double>::iterator it=values.begin(); it!=values.end(); ++it)
      std::cout << it->first << " => " << it->second << '\n';

  string search_for = "BBB";

  cout << "Using lower bound " << endl;
  for (std::map<string,double>::iterator it=values.lower_bound(search_for); it!=values.end(); ++it)
      std::cout << it->first << " => " << it->second << '\n';
  cout << "Using upper bound " << endl;
  for (std::map<string,double>::iterator it=values.upper_bound(search_for); it!=values.end(); ++it)
      std::cout << it->first << " => " << it->second << '\n';

  cout << "Printing only substrings " << endl;
  for (std::map<string,double>::iterator it=values.lower_bound(search_for); it!=values.end(); ++it){
    string key = it->first;
    if (key.compare(0, search_for.size(), search_for) == 0){
      std::cout << it->first << " => " << it->second << '\n';
    }
  }


  exit(0);
  */

  /*
  // testing findMap
  map<int,double> testMap;
  for (uint i=0;i<5;i++){
    pair<int,double> mypair(i,i*2.5);
    testMap.insert(mypair);
  }
  double result = findMap(testMap,3,MSG_NO_MSG);
  double result2 = findMap(testMap,1,MSG_ERROR);
  double result3 = findMap(testMap,7,MSG_DEBUG);
  cout << "Done" << endl;
  map<int, vector <double> > testMap2;
  for (uint i=0;i<5;i++){
    vector <double> myvector;
    for(uint j=0;j<10;j++) {
      myvector.push_back(i*100+j);
    }
    pair<int,vector <double> > mypair2(i,myvector);
    testMap2.insert(mypair2);
  }
  vector <double> resultb = findMap(testMap2,3,MSG_NO_MSG);
  vector <double> resultb2 = findMap(testMap2,1,MSG_ERROR);
```

```
  vector <double> resultb3 = findMap(testMap2,7);
  cout << "Done2" << endl;
  exit(1);
  */



  /*
  // Testing vSum
  vector <int> ivector(5,5);
  vector <double> dvector(5,1.5);
  vector < vector <int> > ivector2;
  vector <vector <double > > dvector2;


  for(uint i=0;i<5;i++){
    ivector2.push_back(ivector);
    dvector2.push_back(dvector);
  }

  int iSum = vSum(ivector);
  int iSum2 = vSum(ivector2[2]);
  double dSum = vSum(dvector);
  double dSum2 = vSum(dvector2[1]);
  int iSum3 = vSum(ivector2);
  double dSum3 = vSum(dvector2);

  cout << "hi there" << endl;
  */

  /*
  // Testing Eigen
  using Eigen::MatrixXd;
  MatrixXd m(2,2);
  m(0,0) = 4;
  m(1,0) = 2.5;
  m(0,1) = -1;
  m(1,1) = m(1,0) + m(0,1);
  std::cout << m << std::endl;
  exit(0);
  */

  /*
  // Test on two different type of partial matching over map values
  testPartMatching2();
  testPartMatching();
  */

  /*
  // -------------------------------------------------------------------
  // Testing how to erase elements from a vector according to conditions
  // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

  vector<string> myvector;
  myvector.push_back("a");
  myvector.push_back("b");
  myvector.push_back("c");
  myvector.push_back("d");
  myvector.push_back("e");

  for (uint i=0; i<myvector.size();i++){
    cout << "i:" << i << " myvector[i]: " << myvector[i] << endl;
    if(myvector[i]== "c" || myvector[i]=="d"){
      cout << " -- TBR: " << "i:" << i << " myvector[i]: " << myvector[i] << endl;
      myvector.erase (myvector.begin()+i);
      i--;
    }
  }

  cout << "Myvector now contains:" << endl;
  for (int i=0; i<myvector.size(); i++) {
    cout << "i: " << i << " myvector[i]: " << myvector[i] << endl;
  }
  exit (0);
  */


}

void
Sandbox::fullTest(){

 /*
  // Getting forest area by each forest type
  vector<int> regIds2 = MTHREAD->MD->getRegionIds(2);
  for(uint r=0;r<regIds2.size();r++){
    int rId = regIds2[r];
```

```
   ModelRegion* reg = MTHREAD->MD->getRegion(regIds2[r]);
   vector <string> fTypes= MTHREAD->MD->getForTypeIds();
   for(uint f=0;f<fTypes.size();f++){
     string ft = fTypes[f];
     forType* FT = MTHREAD->MD->getForType(ft);
     double totalArea = 0.0;
     vector <Pixel*> rpx = MTHREAD->GIS->getAllPlotsByRegion(regIds2[r]);
     for (uint p=0;p<rpx.size();p++){
       Pixel* px = rpx[p];
       totalArea +=  px->getDoubleValue (FT->forLayer, true);
     }
     cout << rId << "\t" << ft << "\t" << totalArea << endl;
   }
 }
 exit(1);
*/

 /*
 // Testing the new getForTypeParents()function
 vector<string>  parents = MTHREAD->MD->getForTypeParents();
 for(uint i=0;i<parents.size();i++){
   vector <string> childIds = MTHREAD->MD->getForTypeChilds(parents[i]);
   vector <int> childPos = MTHREAD->MD->getForTypeChilds_pos(parents[i]);
   double debug = 0.0;
 }
 */

 /*
 // Testing the reg->getArea() functions
 // Actually this need to be run further later, as pixels doesn’t yet have area information
 vector <string> dClasses = MTHREAD->MD->getStringVectorSetting("dClasses");
 vector <string> fTypes= MTHREAD->MD->getForTypeIds();
 ModelRegion* REG = MTHREAD->MD->getRegion(11041);
 cout << "Total ft area: "<< REG->getArea()<< endl;

 for(uint j=0;j<fTypes.size();j++){
   cout << fTypes[j] << "\t" << REG->getArea(fTypes[j]) << "\t" << REG->getArea(j) << endl;
 }
 for(uint j=0;j<fTypes.size();j++){
   cout << fTypes[j] << "\t" << REG->getArea(fTypes[j]) << "\t";
   for(uint u=0;u<dClasses.size();u++){
     cout << REG->getArea(j,u) << " ";
   }
   cout << endl;
 }
 */

 /*
 // Testing getForData() function with no forest id specified
 double vartest= MTHREAD->MD->getForData("forestChangeAreaIncrementsRel",11061,"","",2009);
 cout << vartest << endl;
 exit(0);
 */


 /*
 // Testing the decay model  - ok, passed
 double initialValue = 100;
 double halfLife = 2;
 double years = 0;
 double remStock = MTHREAD->CBAL->getRemainingStock(initialValue, halfLife, years); ///< Apply a single exponential decay model to retrieve the remining stock given the initial stock, the half life and the time passed from stock formation.
 cout << "Remaining stock: " << remStock << endl;
 years = 1;
 remStock = MTHREAD->CBAL->getRemainingStock(initialValue, halfLife, years);
 cout << "Remaining stock: " << remStock << endl;
 years = 5;
 remStock = MTHREAD->CBAL->getRemainingStock(initialValue, halfLife, years);
 cout << "Remaining stock: " << remStock << endl;
 years =10;
 remStock = MTHREAD->CBAL->getRemainingStock(initialValue, halfLife, years);
 cout << "Remaining stock: " << remStock << endl;
 years = 200;
 remStock = MTHREAD->CBAL->getRemainingStock(initialValue, halfLife, years);
 cout << "Remaining stock: " << remStock << endl;
 */

 /*
 // Testing normSample
 // template <typename K> K normSample (const K& avg, const K& stdev, const K& minval=NULL, const K& maxval=NULL)
 // template <typename K> K normSample (const normal_distribution<K>& d, const std::mt19937& gen, const K& minval=NULL, const K& maxval=NULL)
 double avg = 0.8;
 double stdev = 0.2;
 double minval = 0.0;
 double maxval = 1.0;
 double result;

 cout << "Starting first method.." << endl;
 normal_distribution<double> d(avg,stdev);
 std::mt19937 gen = *MTHREAD->gen;
 for (uint i=0;i<1000;i++){
   result = normSample(d, gen, minval, maxval);
   cout << "Result1: " << result << endl;
 }
 cout << "Finished first method and starting second one.." << endl;
 for (uint i=0;i<1000;i++){
   result = normSample(avg, stdev, minval, maxval);
   cout << "Result2: " << result << endl;
 }
 cout << "Finished second method."<< endl;

 exit(0);
 */


  //double disttest = MTHREAD->MD->getProdData("dist",11042,"",DATA_NOW,i2s(11061));
  //cout << disttest << endl;
  //exit(0);


  /*double test = MTHREAD->CBAL->getStock(11061, STOCK_INV);
  //STOCK_INV  -> from inventary source and death trees
  //STOCK_EXTRA  -> from inventary source and death trees
  //STOCK_PRODUCTS -> from products
  cout << "DONE" << endl;
  exit(0);
  */

   /*
   // Testing if forestData can uses other arbitrary elements in the diameter field in order to generalise it
   double test = MTHREAD->MD->getForData("covar",11082,"con_highF","con_highF");
   MTHREAD->MD->setForData(0.1,"covar",11082,"con_highF","con_highF");
   MTHREAD->MD->setForData(0.1,"covar",11061,"con_highF","con_highF",DATA_NOW,true);
   test = MTHREAD->MD->getForData("covar",11082,"con_highF","con_highF");
   test = MTHREAD->MD->getForData("covar",11061,"con_highF","con_highF");
   test = MTHREAD->MD->getForData("covar",11082,"con_highF","");
   cout << test << endl;
   exit(0);
   */

   /*
   // Testing getProdData for the freeDimension
   MTHREAD->MD->setProdData(0.4,"rt",11041,"hardWSawnW",DATA_NOW,true,"11061");
   MTHREAD->MD->setProdData(0.3,"rt",11041,"hardWSawnW",DATA_NOW,true,"11030");
   MTHREAD->MD->setProdData(0.2,"rt",11041,"hardWSawnX",DATA_NOW,true,"11030");
   double debug = MTHREAD->MD->getProdData("rt",11041,"hardWSawnW",DATA_NOW,"11061");
   double debug2 = MTHREAD->MD->getProdData("rt",11041,"hardWSawnW",DATA_NOW);
   cout << debug << "    " << debug2 << endl;
   exit(0);
   */

   /*
   // Testing api to call generic forest type data, parent/child
   cout << "Hello world " << endl;
   cout << MTHREAD->MD->getForData("freq_norm",11041,"broadL","",2040) << endl;
   MTHREAD->MD->setForData(100,"freq_norm",11041,"broadL","",2040);
   cout << MTHREAD->MD->getForData("freq_norm",11041,"broadL","",2040) << endl;
   cout << MTHREAD->MD->getForTypeParentId("broadL_highF")<< endl;
   cout << MTHREAD->MD->getForTypeParentId("con_highF")<< endl;
   exit(0);
   */

   /*
   // Testing for each region how far is the average of the multipliers from 1
   vector<int>  regIds  =    MTHREAD->MD->getRegionIds(2);
   vector <string> ftypes = MTHREAD->MD->getForTypeIds();

   cout << "*** Checking how far is the tpMultiplier far from 1 in each region:" << endl;
   for (int i=0;i< regIds.size();i++){
       ModelRegion* region = MTHREAD->MD->getRegion(regIds[i]);
       vector <Pixel*>  regpixels =   MTHREAD->GIS->getAllPlotsByRegion(regIds[i]);
       if(regpixels.size()==0) continue;
       cout << "*** " << region->getRegLName() << ":  "<< endl;
       for(int ft = 0;ft<ftypes.size();ft++){
           double tot = 0;
           double avg = 0;
           for(int j=0;j<regpixels.size();j++){
             tot += regpixels[j]->getSpModifier(ftypes[ft]);
           }
           avg = tot/regpixels.size();
           cout << ftypes[ft] << ":  " << avg << endl;
       }
   }
   exit(0);
   */

   /*
   // Testing the number of plots in the model
   vector <ModelRegion*> regions = MTHREAD->MD->getAllRegions();
   int total = 0;
   cout << "*** Pixels by region:" << endl;
   for (int i=0;i< regions.size();i++){
       vector <Pixel*>  regpixels =   MTHREAD->GIS->getAllPlotsByRegion(*regions[i]);
       cout << regions[i]->getRegLName() << ":  " << regpixels.size() << endl;
       total += regpixels.size() ;
   }
   cout << "** Total:  " << total << endl;
   exit(0);
   */

 /*
 // Testing the new random distributions. Requires the pointer MTHREAD->gen to be initialised,
 // so this test can't run in basic test.
 std::normal_distribution<double> d(100000,3); // default any how to double
 for(int n=0; n<20; ++n) {
   double x = d(*MTHREAD->gen);
   int i = round(d(*MTHREAD->gen));
   cout << i << ';' << 1 << endl;
 }
 exit (0);
 */

 /*
 // Testing I have correctly the info about world price !!!
 // yes, it seems ok here !!!
 int firstYear = MTHREAD->MD->getIntSetting("initialYear");
 int initialOptYear= MTHREAD->MD->getIntSetting("initialOptYear");
 int simulationYears = MTHREAD->MD->getIntSetting("simulationYears");
 int WL2 = MTHREAD->MD->getIntSetting("worldCodeLev2");
 vector <string> priProducts = MTHREAD->MD->getStringVectorSetting("priProducts");
 vector <string> secProducts = MTHREAD->MD->getStringVectorSetting("secProducts");
 vector <string> allProducts = priProducts;
 allProducts.insert( allProducts.end(), secProducts.begin(), secProducts.end() );

 for(uint i=0;i<allProducts.size();i++){
   for(int y=firstYear; y<initialOptYear+simulationYears; y++){
     double pw = MTHREAD->MD->getProdData("pl",WL2,allProducts[i],y);
     cout << allProducts[i] << "  " << y << "  " << pw << endl;
   }
 }
 exit (0);
 */

 /*
 // testing Pixel::getMultiplier (const string& multiplierName, const string& forName, int year)
 Pixel* px = MTHREAD->GIS->getPixel(0);
 double debug1 = px->getMultiplier("tp_multiplier","broadL_highF",2012);
 double debug2 = px->getMultiplier("tp_multiplier","broadL_highF",2008);
 double debug3 = px->getMultiplier("tp_multiplier","broadL_highF",2009);
 double debug4 = px->getMultiplier("tp_multiplier","broadL_highF",2010);
 double debug5 = px->getMultiplier("mortCoeff_multiplier","broadL_highF",2012);
 double debug6 = px->getMultiplier("mortCoeff_multiplier","con_copp",2012);
 double debug7 = px->getMultiplier("blaaaa","broadL_highF",2012);

 double debug10 = debug1;
*/

 /*
 // testing reading a directory
 string dir = MTHREAD->MD->getBaseDirectory()+MTHREAD->MD->getStringSetting("spatialDataSubfolder");
 vector<string> files = vector<string>();

 MTHREAD->MD->getFilenamesByDir (dir,files, ".grd");

 for (unsigned int i = 0;i < files.size();i++) {
   cout << files[i] << endl;
 }
 */

 /*
 // testing ModelData:: ModelData::calculateAnnualisedEquivalent(double amount_h, int years_h)
 cout << "Value:" << MTHREAD->MD->calculateAnnualisedEquivalent(500.,4) << endl;
 cout << "Value:" << MTHREAD->MD->calculateAnnualisedEquivalent(500.,30) << endl;
 cout << "Value:" << MTHREAD->MD->calculateAnnualisedEquivalent(107.035040105,10) << endl;
 cout << "Value:" << MTHREAD->MD->calculateAnnualisedEquivalent(8.91507,1) << endl;
 cout << "Done" << endl;
 exit(0);
```

```
  */

  /*
  // testing snprintf
  vector <int> myintegers;
  vector <double> mydoubles;
  char  szResult[24];

  myintegers.push_back(1);
  myintegers.push_back(202);
  myintegers.push_back(3003);
  myintegers.push_back(400004);
  myintegers.push_back(50000005);
  myintegers.push_back(6000000006);
  mydoubles.push_back(1.1234567890);
  mydoubles.push_back(12345678.9);
  mydoubles.push_back(12345678.90123456);
  mydoubles.push_back(6000000006.123456789012);
  for(uint i=0;i<myintegers.size();i++){
    snprintf ( szResult, sizeof(szResult), "%d", myintegers[i] );  // "safe" version
    cout << "int/string: " << myintegers[i] << "  /  " << szResult << endl;
  }
  for(uint i=0;i<mydoubles.size();i++){
      snprintf ( szResult, sizeof(szResult), "%f", mydoubles[i] );  // "safe" version
    cout << "double/string: " << mydoubles[i] << "  /  " << szResult << endl;
  }
  exit(0);
  */

  /*
  // testing stod() ..
  // this is giving different results if gui or console mode !!
  vector<string> numbers;
  numbers.push_back("123.1234567890");
  numbers.push_back("123.1234");
  numbers.push_back("123,1234567890");
  numbers.push_back("123,1234");
  double outd;
  for(uint i=0;i<numbers.size();i++){
    try {
      outd =  stod(numbers[i]);
      cout << "Conversion passed: " << numbers[i] << "  -  " << outd << endl;
    } catch (...) {
      cout << "Conversion DID NOT passed: " << numbers[i] << "  -  " <<endl;
    }
  }
  exit(0);
  */

/*
// -----------------------------------
// Testing makeKeyProdData() and unpackKeyProdData()
  string parName = "za";
  int regId = 20000;
  string prod = "wood";
  string freeDim = "";
  string key = MTHREAD->MD->makeKeyProdData(parName,i2s(regId),prod,freeDim);
  cout << "key: " << key << endl;
  MTHREAD->MD->unpackKeyProdData(key,parName,regId,prod,freeDim);
  cout << "parName: " << parName << endl;
  cout << "regId: " << regId << endl;
  cout << "prod: " << prod << endl;
  cout << "freeDim: " << freeDim << endl;
  exit(0);
*/

/*
// ---------------------------------------------
// checking the functions dataMapCheckExist() and dataMapGetValue() works well
typedef map<string, vector <double> > DataMap;
typedef pair<string, vector <double> > DataPair;

vector <double> abaa (5, 1.);
vector <double> abcc (5,10);
vector <double> anbb (5,100);
vector <double> andd (5,5);
vector <double> anff (5,3);
vector <double> ag (5,2);
vector <double> agii (5,7);



DataMap dM;
dM.insert(DataPair("abaa", abaa));
dM.insert(DataPair("abcc", abcc));
dM.insert(DataPair("anbb", anbb));
dM.insert(DataPair("andd", andd));
dM.insert(DataPair("anff", anff));
dM.insert(DataPair("ag", ag));
dM.insert(DataPair("agii", agii));

vector<string> tests;
tests.push_back("ab");
tests.push_back("anbb");
tests.push_back("ane");
tests.push_back("an");
tests.push_back("ac");
tests.push_back("ag");
tests.push_back("agii");
tests.push_back("al");


bool found;
double value;

for(uint i=0;i<tests.size();i++){
  found = MTHREAD->MD->dataMapCheckExist(dM, tests[i]);
  value = MTHREAD->MD->dataMapGetValue(dM, tests[i],2010);
  cout << tests[i] << ": " << b2s(found) << " value: "<< value << endl;
}

exit(0);
*/


  /*
  // testing how to search on a vector using the find algorithm

  vector<string> names;
  names.push_back("pippo");
  names.push_back("topolino");
  names.push_back("minni");
  names.push_back("paperino");

  string toSearch1 = "minni";
  string toSearch2 = "zio paperone";

  if( find(names.begin(), names.end(), toSearch1)!= names.end() ){
    cout << "minni trovata" << endl;
  }
    if( find(names.begin(), names.end(), toSearch2)!= names.end() ){
    cout << "zio paperone trovato" << endl;
  }
  cout << "test on find ended." << endl;
  exit(0);
  */

// --------------------------------------------------------------------


  /*
  int a;
  a = getSetting<int>("myIntData", TYPE_INT);

  string b;
  b = getSetting<string>("myStringData", TYPE_STRING);

  bool c;
  c = getSetting<bool>("myBoolData", TYPE_BOOL);


  cout << "A is: " << a << endl;

  cout << "B is: " << b << endl;

  cout << "C is: " << c << endl;

  //vector<string> getVectorSetting <string> ("test", TYPE_STRING);
  //template <class T> vector <T> getVectorSetting(string name_h, int type);

  //vector <string> myStrings = getVectorSetting <vector<string> > ("test", TYPE_STRING);

  string s = GccTest("test");
  int i = GccTest("test");
  vector <int> iVector = GccTest("test");

  for (int i=0; i< iVector.size(); i++){
    cout << "iVector: " << iVector.at(i) << endl;
  }
  */

  // --------------------------------------------------------------------

  /* // I learned: how to access elements - both objects and pointers - of a vector using pointers
```

```
// testing how to operate with iterators over a pointer element in an array:

cout << "Starting iterator test..." << endl;

TestStructure a,b,c,d;
a.i=0; b.i=1; c.i=2; d.i=3;
TestStructure* ap;
TestStructure* bp;
TestStructure* cp;
TestStructure* dp;

ap = &a;
bp = &b;
cp = &c;
dp = &d;

vector <TestStructure>  objects;
vector <TestStructure*> pointers;

objects.push_back(a);
objects.push_back(b);
objects.push_back(c);
objects.push_back(d);

pointers.push_back(ap);
pointers.push_back(bp);
pointers.push_back(cp);
pointers.push_back(dp);

vector<TestStructure>::iterator pi;
vector<TestStructure*>::iterator pp;

 //ok it works
//for ( pi = objects.begin() ; pi != objects.end();){
//  if(pi->i==2){
//    objects.erase(pi);
//  }
//  else {
//    ++pi;
//  }
//}

//for (int j=0;j<objects.size();j++){
//  cout << j << " object is: " << objects[j].i << endl;
//}


// works as well ;-))
for ( pp = pointers.begin() ; pp != pointers.end();){
  if( (*pp)->i==2){
    //delete (*pp);
    pointers.erase(pp);
  }
  else {
    ++pp;
  }
}

for (int j=0;j<pointers.size();j++){
  cout << j << " pointers is: " << pointers[j]->i << endl;
}

// c is not destructed if we don't explicitelly call delete over the pointer...
cout << c.i << endl; // this go in seg-frag if we call delete (*pp)..
*/

// ------------------------------------------------------------------
/* test on how to remove from a map.. deletable
map <int, string> test;
test.insert(pair<int, string>(2, "pippo"));
test.insert(pair<int, string>(1, "pluto"));
test.insert(pair<int, string>(5, "minni"));
test.insert(pair<int, string>(3, "topolino"));


map <int, string>::iterator p;
p=test.find(3);
if(p != test.end()){
  cout << p->second <<endl;
  test.erase(p);
}
else {
  cout << "not found " << endl;
}

map <int, string>::iterator p2;
p2=test.find(3);
```

```
  if(p2 != test.end()){
    cout << p2->second <<endl;
    test.erase(p2);
  }
  else {
    cout << "not found " << endl;
  }
  */

   /*vector<int> test;
   for (int i=0;i<5;i++) test.push_back(i);
   cout << "test.." << endl;
  for (uint i=0;i<test.size();i++){
     cout << "Test "<<i<<": "<<test.at(i) << endl;
  }
  //test.erase(2);

  vector<int>::iterator p;
  for ( p = test.begin() ; p != test.end();){
    if(*p == 1 || *p == 2 || *p==4){
      test.erase(p);
    }
    else {
      ++p;
    }
  }


  for (uint i=0;i<test.size();i++){
     cout << "Test "<<i<<": "<<test.at(i) << endl;
  }

//  test.erase(remove_if(test.begin(), test.end(), FindMatchingString(&fs))

//  for (int i=0;i<test.size();i++) cout << "TEST: "<<i<< " " << test.at(i) << endl;
  */

  /*
  // On this test I am showing how to "move" one pointer from a vector of pointers to an other one. The real case is used to move Agent_farmer* pointers from the managedAgents vector to the removedVector.

  double* myDouble1 = new double(1);
  double* myDouble2 = new double(2);
  double* myDouble3 = new double(3);

  vector <double*> origin;
  vector <double*> destination;

  origin.push_back(myDouble1);
  origin.push_back(myDouble2);
  origin.push_back(myDouble3);

  cout << "MyDouble2: "<< *myDouble2 << endl;
  vector<double*>::iterator doublePIterator;

  for (int i=0;i<origin.size();i++){
    cout << i << " origin is: " << *origin[i] << endl;
  }

  for ( doublePIterator = origin.begin() ; doublePIterator !=origin.end();){
    if(*doublePIterator == myDouble2){
      destination.push_back(myDouble2);
      origin.erase(doublePIterator);
    }
    else {
      ++doublePIterator;
    }
  }

  for (int i=0;i<origin.size();i++){
    cout << i << " origin is now: " << *origin[i] << endl;
  }

  for (int i=0;i<destination.size();i++){
    cout << i << " destination is: " << *destination[i] << endl;
  } */

  // --------------------------------------------------------------------
  /*
  // Test on how to return a vector of pointers from a member vector of data
  TestStructure a,b,c,d;
  a.i=0; b.i=1; c.i=2; d.i=3;
  testVector.push_back(a);
  testVector.push_back(b);
  testVector.push_back(c);
  testVector.push_back(d);

  vector<TestStructure*>  myVector=getTestStructure();

  for(uint i=0;i<myVector.size();i++){
    msgOut(MSG_DEBUG, i2s(myVector[i]->i));
  }
  */

  /*
  // Deleting an object and inserting a new one on a vector of objects.. it doesn't works.. problems with the last element..
  vector<BasicData>::iterator p;
  for(p=programSettingsVector.begin();p!=programSettingsVector.end();p++){
    if(p->name == SETT.name){
      programSettingsVector.erase(p);
      programSettingsVector.insert(p,1,SETT);
      cout << SETT.name <<endl;
      break;
    }
  }
  */

  /*double test = -987654321.987654321;
  double result;
  result = fabs(test);
  cout << "Test: "<< result << endl;*/


  /*
  // Testing the zip library:

  cout <<"Hello world Zip!" << endl;

  QString file = "data/testInput.ods";
  QString out = "data/tempInput";
  QString pwd = "";
  if (!QFile::exists(file))
  {
    cout << "File does not exist." << endl << endl;
    //return false;
  }

  UnZip::ErrorCode ec;
  UnZip uz;

  if (!pwd.isEmpty())
    uz.setPassword(pwd);

  ec = uz.openArchive(file);
  if (ec != UnZip::Ok)
  {
        //cout << "Failed to open archive: " << uz.formatError(ec).toAscii().data() << endl << endl; // Qt4
        cout << "Failed to open archive: " << uz.formatError(ec).toLatin1().data() << endl << endl;  // Qt5
    //return false;
  }

  ec = uz.extractAll(out);
  if (ec != UnZip::Ok)
  {
        //cout << "Extraction failed: " << uz.formatError(ec).toAscii().data() << endl << endl; // Qt4
        cout << "Extraction failed: " << uz.formatError(ec).toLatin1().data() << endl << endl;  // Qt5
    uz.closeArchive();
    //return false;
  }
  */

  /*
  // How to : delete an element from an array from its position
  cout << "How to : delete an element from an array from its position" << endl;

  vector <string> headers;
  vector < vector <string> > records;
  vector <string> firstrecord;
  vector <string> secondrecord;
  records.push_back(firstrecord);
  records.push_back(secondrecord);

  headers.push_back("a");
  headers.push_back("b");
  headers.push_back("");
  headers.push_back("d");
  headers.push_back("e");
  headers.push_back("");

  records[0].push_back("0");
  records[0].push_back("1");
  records[0].push_back("2");
  records[0].push_back("3");
  records[0].push_back("4");
  records[0].push_back("5");
  records[1].push_back("00");
  records[1].push_back("11");
  records[1].push_back("22");
  records[1].push_back("33");
  records[1].push_back("44");
  records[1].push_back("55");

  for (int i=headers.size()-1;i>=0;i--){
    if(headers[i] == ""){
      headers.erase(headers.begin()+i);
      for (int j=0;j<records.size();j++){
        records[j].erase(records[j].begin()+i);
      }
    }
  }
  for(uint i=0;i<headers.size();i++){
    cout << headers.at(i) << " - " << records[0].at(i) << " - " << records[1].at(i) << endl;
  }
  cout << "done!" << endl;
  */

  //testThreads();
  /*vector<double> numbers;
  double cumNumbers = 0.00;
  numbers.push_back(0.40);
  numbers.push_back(0.10);
  numbers.push_back(0.20);
  numbers.push_back(0.08);
  numbers.push_back(0.22);

  for (uint i=0;i<numbers.size();i++){
    cumNumbers += numbers[i];
  }

  if (cumNumbers <= 0.99999999 || cumNumbers >= 1.00000001) {
    cout <<"Bastardo!"<<endl;
  } else {
    cout <<"qui funzia!"<<endl;
  }*/

}

template <class T> vector <T> getVectorSetting(string name_h, int type) {

    vector <string> myStringDatas;
    myStringDatas.push_back("aaaaa");
    myStringDatas.push_back("bbbbb");
    myStringDatas.push_back("ccccc");
    vector <T> xVector;

    for (int i=0;i<myStringDatas.size();i++){
      istringstream iss(myStringDatas[i]);
      T x;
      iss >> x;
      xVector.push_back(x);
    }

    return xVector;
}




template <class T> T
Sandbox::getSetting(string name_h, int type){

  string myIntData;
  myIntData = "34";
  string myStringData;
  myStringData = "abcdefg";

  string myBoolData;
  myBoolData = "false";

  if(type==TYPE_INT){
    istringstream iss(myIntData);
    T x;
    iss >> x;
    return x;
  }

  if(type==TYPE_STRING){
    istringstream iss(myStringData);
    T x;
    iss >> x;
    return x;
  }

  if(type==TYPE_BOOL){
    string tempBoolString;
    if (myBoolData == "1" || myBoolData == "true" || myBoolData == "True" || myBoolData == "TRUE" || myBoolData == "vero" || myBoolData == "Vero"|| myBoolData == "VERO"){
      tempBoolString = "1";
    }
    else if (myBoolData == "0" || myBoolData == "false" || myBoolData == "False" || myBoolData == "FALSE" || myBoolData == "falso" || myBoolData == "falso"|| myBoolData == "FALSO"){
      tempBoolString = "0";
    }
    else {
      msgOut(MSG_CRITICAL_ERROR, "Impossible conversion of "+myBoolData+" to bool!. Aborted.");
    }
    istringstream iss(tempBoolString);
    T x;
    iss >> x;
    return x;
  }


}

template<typename T> T
Sandbox::test2(const std::string& s) {
  std::istringstream iss(s);
  T x;
  iss >> x;
  return x;
}


vector <TestStructure*>
Sandbox::getTestStructure(){
  vector <TestStructure*> toReturn;
  for (uint i=0;i<testVector.size();i++){
    //TestStructure* tempTest = new TestStructure;
    toReturn.push_back(&testVector[i]);
  }
  return toReturn;

}



void
Sandbox::testThreads(){

        /*
        PSEUDOCODE
        - attivo i vari thread
        - per ogni closestAgent itero fra i vari thread e se "è libero" gli assegno il closestAgent
        - quando ho finito i closestAgent aspetto che tutti i threads abbiano finito il lavoro
        - chiudo i threads
        - vado avanti
        */
        int nAgents= 50;
        vector<TestStructure*> myAgents;
        vector<double> myResults (nAgents, (double) 0);
        //int nThreads = MTHREAD->MD->getIntSetting("nThreads");
        int nThreads= 5;

        for (int i=0; i < nAgents; i++){
          TestStructure* myAgent = new TestStructure;
          myAgent->i = i;
          myAgent->random =  (0+( (double)rand() / ((double)(RAND_MAX)+(double)(1)) )*(10-0+1))/ (double)100;
          myAgents.push_back(myAgent);
        }

        vector <testThread*> myThreads ;

        for (int i=0; i < nThreads; i++){
          testThread* myThread = new testThread;
          myThreads.push_back(myThread);
        }

        for (uint i=0;i<myAgents.size();i++){
          bool assigned = false;
          while(!assigned) {
            for (uint j=0;j<myThreads.size();j++){
              if (!myThreads[j]->isRunning()){
                cout << "Assigning agent " << i << " to thread " << j << endl;
                myThreads[j]->assignJob(myAgents[i]);
                myThreads[j]->start();
                assigned = true;
                break;
              }
              else {
                cout << "Thread " << j << " is busy" << endl;
              }
            }
          }
        }
        /*
        volatile bool somethingStopping = true;
        while (somethingStopping){
          somethingStopping = false;
          for (uint i=0;i<myThreads.size();i++){
            if(myThreads[i]->isRunning()){
              somethingStopping = true;
              //cout << "somethingStopping is true" << endl;
            }
          }
        }

        if (somethingStopping) {
          cout << "somethingStopping is true" << endl;
        }
        else {
          cout << "somethingStopping is false" << endl;
        }
        cout << "pinco pallo sono nel mezzo dei threads..."<<endl;
        */
        for (int i=0; i < nThreads; i++){
          myThreads[i]->wait();
        }


        for (int i=0; i < nThreads; i++){
          delete myThreads[i];
        }

        for (uint i=0;i<myAgents.size();i++){
          //cout <<myAgents[i]->cachedOffer<<endl;

          double random =  (0+( (double)rand() / ((double)(RAND_MAX)+(double)(1)) )*(10-0+1))/ (double)100;

          // important !
          // for random integer see also std::uniform_int_distribution :
          // http://stackoverflow.com/questions/7780918/stduniform-int-distributionint-range-in-g-and-msvc
          // in regmas:
          // int randomRed = int (50+( (double)rand() / ((double)(RAND_MAX)+(double)(1)) )*(255-50+1)); // randomRed is [50,255] Don't use "randomNumber % range" !!

          //cout <<random<<endl;
        }

        //thread1.stop();
        cout << "FINITO"<<endl;


}

testThread::testThread(){

}

void
testThread::run(){

  cout << agent->i << endl;

  double randChange =  (0+( (double)rand() / ((double)(RAND_MAX)+(double)(1)) )*(10-0+1))/ (double)100; // rand() must be not thread safe !!!!

  int justn = 10000;
  vector <double> takeTimeVector (justn, 0);
  for (int i =0; i< justn;i++){
    takeTimeVector.at(i)=i*2;
  }
  agent->cachedOffer = agent->random;
}

void
testThread::assignJob(TestStructure* agent_h){
  agent = agent_h;
  agent->cachedOffer = 0;
}

void
```

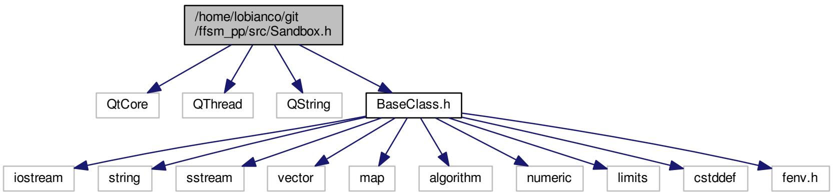
```
Sandbox::testIpopt(){


    using namespace Ipopt;

      // Create a new instance of your nlp
      //  (use a SmartPtr, not raw)
      SmartPtr<TNLP> mynlp = new Ipopt_nlp_problem_debugtest();

      // Create a new instance of IpoptApplication
      //  (use a SmartPtr, not raw)
      // We are using the factory, since this allows us to compile this
      // example with an Ipopt Windows DLL
      SmartPtr<IpoptApplication> app = IpoptApplicationFactory();

      // Change some options
      // Note: The following choices are only examples, they might not be
      //       suitable for your optimization problem.
      app->Options()->SetNumericValue("tol", 1e-7);
      app->Options()->SetStringValue("mu_strategy", "adaptive");
      app->Options()->SetStringValue("output_file", "ipopt.out");
      //app->Options()->SetStringValue("hessian_approximation", "limited-memory");
      //app->Options()->SetStringValue("derivative_test", "second-order");
      //app->Options()->SetStringValue("derivative_test_print_all", "yes");


      // The following overwrites the default name (ipopt.opt) of the
      // options file
      // app->Options()->SetStringValue("option_file_name", "hs071.opt");

      // Intialize the IpoptApplication and process the options
      ApplicationReturnStatus status;
      status = app->Initialize();
      if (status != Solve_Succeeded) {
        std::cout << std::endl << std::endl << "*** Error during initialization!" << std::endl;
        //return (int) status; // here the abort
      }

      // Ask Ipopt to solve the problem
      status = app->OptimizeTNLP(mynlp);

      if (status == Solve_Succeeded) {
        std::cout << std::endl << std::endl << "*** The problem solved!" << std::endl;
      }
      else {
        std::cout << std::endl << std::endl << "*** The problem FAILED!" << std::endl;
      }

}

int
Sandbox::testAdolc(){

  using namespace Ipopt;
 // Create an instance of your nlp...
   SmartPtr<TNLP> myadolc_nlp = new MyADOLC_NLP();
  //SmartPtr<TNLP> myadolc_nlp = new MyADOLC_sparseNLP();

  // Create an instance of the IpoptApplication
  SmartPtr<IpoptApplication> app = new IpoptApplication();

  // Initialize the IpoptApplication and process the options
  ApplicationReturnStatus status;
  status = app->Initialize();
  if (status != Solve_Succeeded) {
    printf("\n\n*** Error during initialization!\n");
    return (int) status;
  }

  status = app->OptimizeTNLP(myadolc_nlp);

  if (status == Solve_Succeeded) {
    // Retrieve some statistics about the solve
    Index iter_count = app->Statistics()->IterationCount();
    printf("\n\n*** The problem solved in %d iterations!\n", iter_count);

    Number final_obj = app->Statistics()->FinalObjective();
    printf("\n\n*** The final value of the objective function is %e.\n", final_obj);
  }

  return (int) status;
}

// ------------------------------------------------------------------
// How to partial matching the key of a map

/*TStrStrMap::iterator
```

```
Sandbox::FindPrefix(const TStrStrMap& map, const string& search_for) {
    TStrStrMap::iterator i = map.lower_bound(search_for);
    if (i != map.end()) {
        const string& key = i->first;
        if (key.compare(0, search_for.size(), search_for) == 0) // Really a prefix?
            return i;
    }
    return map.end();
}
*/

/*
void
Sandbox::testSearchMap(const TStrStrMap& map, const string& search_for) {
   cout << search_for;
    TStrStrMap::iterator i = FindPrefix(map, search_for);
    if (i != map.end())
        cout << '\t' << i->first << ", " << i->second;
    cout << endl;

}
*/

void
Sandbox::testSearchMap(const TStrStrMap& map, const string& search_for) {
  TStrStrMap::const_iterator i = map.lower_bound(search_for);
  for(;i != map.end();i++){
    const string& key = i->first;
    if (key.compare(0, search_for.size(), search_for) == 0) {// Really a prefix?
        cout << i->first << ", " << i->second << endl;
    } else {
      break;
    }
  }

}


void
Sandbox::testPartMatching(){

    TStrStrMap tMap;

    tMap.insert(TStrStrPair("John", "AA"));
    tMap.insert(TStrStrPair("Mary", "BBB"));
    tMap.insert(TStrStrPair("Mother", "A"));
    tMap.insert(TStrStrPair("Moliere", "D"));
    tMap.insert(TStrStrPair("Marlon", "C"));

    testSearchMap(tMap, "Marl");
    testSearchMap(tMap, "Mo");
    testSearchMap(tMap, "ther");
    testSearchMap(tMap, "Mad");
    testSearchMap(tMap, "Mom");
    testSearchMap(tMap, "Perr");
    testSearchMap(tMap, "Jo");

  exit(0);
    return;
}

void
Sandbox::testSearchMap2(const TStrStrMap& map_h, const string& search_for)
 {
  TStrStrMap::const_iterator i = map_h.upper_bound(search_for);
  if(i!= map_h.begin())  i--;
  const string& key = i->first;
  string search_base = search_for.substr(0,search_for.size()-4);
  if (key.compare(0, search_base.size(), search_base) == 0){
    cout << "MATCH: " << search_for <<", "<< i->first << ", " << i->second << endl;
  } else {
    cout << "NOTM:  " << search_for <<", "<< i->first << endl;
  }

}

void
Sandbox::testPartMatching2(){

    TStrStrMap tMap;


  tMap.insert(TStrStrPair("mortCoeff_multiplier#broadL_highF##2005", "2005"));
  tMap.insert(TStrStrPair("regLev_1", "-9999"));
  tMap.insert(TStrStrPair("regLev_2", "-9999"));
  tMap.insert(TStrStrPair("tp_multiplier#broadL_copp##2005", "-9999"));
  tMap.insert(TStrStrPair("tp_multiplier#broadL_highF##2005", "50"));
  tMap.insert(TStrStrPair("tp_multiplier#broadL_highF##2010", "2010"));
  tMap.insert(TStrStrPair("tp_multiplier#broadL_mixedF##2005", "-9999"));
  tMap.insert(TStrStrPair("tp_multiplier#con_copp##2005", "-9999"));
  tMap.insert(TStrStrPair("tp_multiplier#con_highF##2005", "-9999"));
  tMap.insert(TStrStrPair("tp_multiplier#con_mixedF##2005", "aa"));

  TStrStrMap::const_iterator i;

  for(i=tMap.begin();i!=tMap.end();i++){
    cout << i->first << ", " << i->second << endl;
  }
  cout << endl;

    testSearchMap2(tMap, "mortCoeff_multiplier#broadL_highF##2006");
    testSearchMap2(tMap, "tp_multiplier#broadL_highF##2008");
    testSearchMap2(tMap, "aaaaaa");
    testSearchMap2(tMap, "zzzzzz");

  exit(0);
    return;
}


```

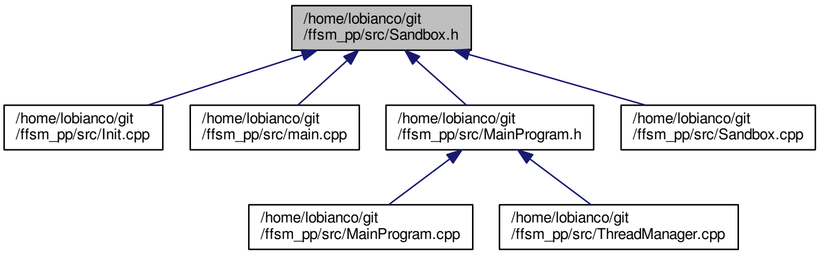
5.125 /home/lobianco/git/ffsm_pp/src/Sandbox.h File Reference

```
#include <QtCore>
#include <QThread>
#include <QString>
#include "BaseClass.h"
```

Include dependency graph for Sandbox.h:



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



Classes

- class Sandbox
- struct TestStructure
- class testThread

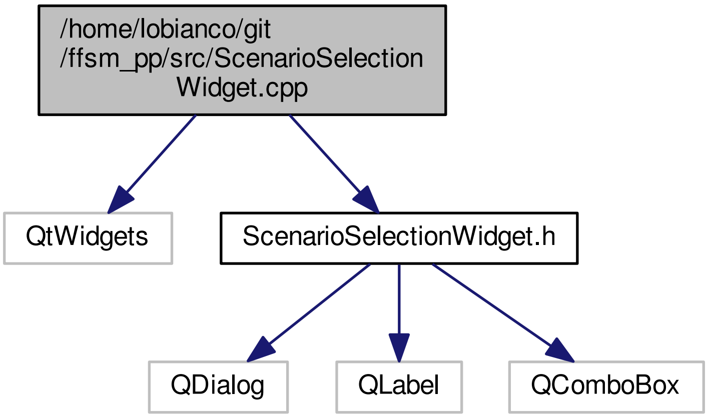
5.126 Sandbox.h

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#ifndef SANDBOX_H
#define SANDBOX_H

#include <QtCore>
#include <QThread>
#include <QString>

#include "BaseClass.h"


/**
This is a test class used when I need to test some C++ or other library functions. It can safelly be
 removed by the project if needed.

  @author Antonello Lobianco <antonello@regmas.org>
*/

struct TestStructure;

class Sandbox : public BaseClass {

public:
                             Sandbox(ThreadManager* MTHREAD_h);
               Sandbox();
                            ~Sandbox();

  template <class T> T       getSetting(string name_h, int type);
  template <class T> vector <T> getVectorSetting(string name_h, int type);
  template <class T> T        test2(const std::string& s); // e.g. int x = test<int>("123");
  void                       printAString(string what){cout << "You printed: "<< what << endl;};
  vector <TestStructure*>    getTestStructure();
  void                       testThreads();
  void                       basicTest(); ///< Simple tests that doesn’t require anything else (are
 encapsulated) and so they can be run at the beginning of the program. Normally empty
  void                       fullTest(); ///< Tests that require a full sandbox object including
 MTHREAD. Normally empty
  void                       testIpopt();
  int                        testAdolc();
  void                       testPartMatching(); ///< How to partial matching the key of a
 map
  void                       testPartMatching2(); ///< How to partial matching the key of a
 map

private:
  vector <TestStructure>  testVector;
  // How to partial matching the key of a map
//  map<string, string>::iterator FindPrefix(const map<string, string>& map, const string& search_for);
  void                         testSearchMap(const map<string, string>& map, const string&
search_for);
  void                         testSearchMap2(const map<string, string>& map_h, const string&
search_for);
};

struct TestStructure {

  int                                         i;
  string                                      s;
  double                            cachedOffer;
  double                                 random;
};


class testThread : public QThread {
  Q_OBJECT

public:
  testThread();
  void                assignJob(TestStructure* agent_h);

protected:
  void run();

private:
  volatile TestStructure* agent;
};



#endif
```

5.127 /home/lobianco/git/ffsm_pp/src/ScenarioSelectionWidget.cpp File Reference

```
#include <QtWidgets>
#include "ScenarioSelectionWidget.h"
```

Include dependency graph for ScenarioSelectionWidget.cpp:


/home/lobianco/git/ffsm_pp/src/ScenarioSelectionWidget.cpp
QtWidgets
ScenarioSelectionWidget.h
QDialog
QLabel
QComboBox

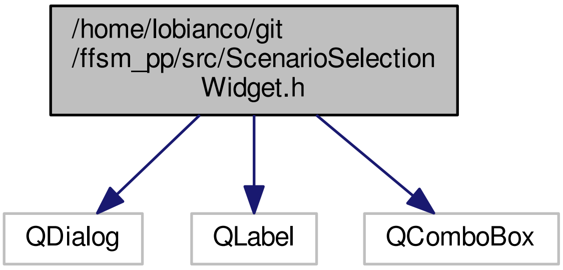
5.128 ScenarioSelectionWidget.cpp

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distributed together    *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/

//#include <QtGui>    // Qt4
#include <QtWidgets> // Qt5


#include "ScenarioSelectionWidget.h"

ScenarioSelectionWidget::ScenarioSelectionWidget(QWidget * parent): QDialog(parent) {

  label = new QLabel(tr("Select the scenario you want to run..."));
  scenarioSelector = new QComboBox();
  QVBoxLayout *mainLayout = new QVBoxLayout;
  mainLayout->addWidget(label);
  mainLayout->addWidget(scenarioSelector);
  setLayout(mainLayout);
  setWindowTitle(tr("Scenario selection"));
  setFixedHeight(sizeHint().height());

  //connect(scenarioSelector, SIGNAL( activated(const QString&)), this, SLOT( processSelectedScenario(const QString &)    ));
  //connect(scenarioSelector, SIGNAL( activated(const QString&)), this, SLOT( close()));

}

ScenarioSelectionWidget::~ScenarioSelectionWidget(){
}


void
ScenarioSelectionWidget::receiveScenarioOptions(const QVector<QString> &scenarios_h){
  scenarioSelector->clear();
  for (uint i=0; i< scenarios_h.size();i++){
    scenarioSelector->addItem(scenarios_h.at(i));
  }
  //scenarioSelector->setFocus(); // may be not visible, no effect!
  //scenarioSelector->grabMouse();
  //scenarioSelector->grabKeyboard();
}

/*
void
ScenarioSelectionWidget::processSelectedScenario(const QString &scenario_h){
  emit selectedScenarioName(scenario_h);
}

*/

```

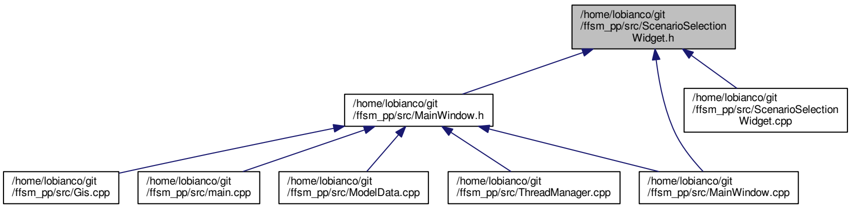
5.129 /home/lobianco/git/ffsm_pp/src/ScenarioSelectionWidget.h File Reference

```
#include <QDialog>
#include <QLabel>
#include <QComboBox>
```

Include dependency graph for ScenarioSelectionWidget.h:



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



Classes

- class ScenarioSelectionWidget

5.130 ScenarioSelectionWidget.h

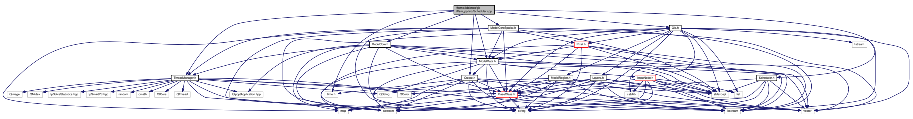
```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#ifndef SCENARIOSELECTIONWIDGET_H
#define SCENARIOSELECTIONWIDGET_H

#include <QDialog>
```

```
#include <QLabel>
#include <QComboBox>
//#include <QtGui>

class QComboBox;

/**
Simple widget to show the available scenarios so that the user can choose one.

  @author Antonello Lobianco <antonello@regmas.org>
*/
class ScenarioSelectionWidget: public QDialog{
  Q_OBJECT

public:
  ScenarioSelectionWidget(QWidget *parent = 0);
  void receiveScenarioOptions(const QVector<QString> &scenarios_h);
  QComboBox                  *scenarioSelector;

private:
  QLabel                                *label;


  ~ScenarioSelectionWidget();

/*
signals:
  void selectedScenarioName(const QString &scenario_h);

public slots:
  void processSelectedScenario(const QString &scenario_h);
*/

};

#endif
```

5.131 /home/lobianco/git/ffsm_pp/src/Scheduler.cpp File Reference

```
#include "time.h"
#include "Scheduler.h"
#include "ThreadManager.h"
#include "Output.h"
#include "ModelData.h"
#include "Gis.h"
#include "ModelCore.h"
#include "ModelCoreSpatial.h"
```

Include dependency graph for Scheduler.cpp:



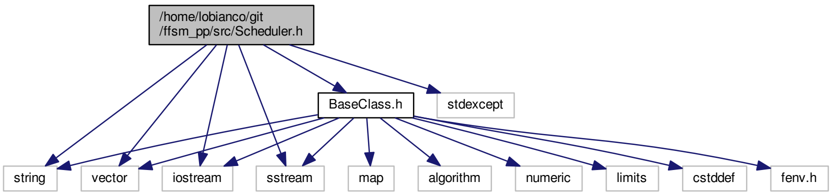
5.132 Scheduler.cpp

```
/****************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière                *
 *   http://ffsm-project.org                                                *
 *                                                                          *
 *   This program is free software; you can redistribute it and/or modify   *
 *   it under the terms of the GNU General Public License as published by   *
 *   the Free Software Foundation; either version 3 of the License, or      *
 *   (at your option) any later version, given the compliance with the      *
 *   exceptions listed in the file COPYING that is distribued together      *
 *   with this file.                                                        *
 *                                                                          *
 *   This program is distributed in the hope that it will be useful,        *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of         *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the          *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#include "time.h"

#include "Scheduler.h"
#include "ThreadManager.h"
#include "Output.h"
#include "ModelData.h"
#include "Gis.h"
#include "ModelCore.h"
#include "ModelCoreSpatial.h"

Scheduler::Scheduler(ThreadManager* MTHREAD_h){
  MTHREAD=MTHREAD_h;
  iteration=0;
}

Scheduler::~Scheduler(){
}

void
Scheduler::run(){

  int initialYear           = MTHREAD->MD->getIntSetting("initialYear");
  int initialSimulationYear = MTHREAD->MD->getIntSetting("initialOptYear");
  int preSimulationYears = initialSimulationYear-initialYear;
  for (int it=preSimulationYears;it<MTHREAD->MD->getIntSetting("simulationYears")+preSimulationYears;it++){
    iteration = it;
    year = iteration+MTHREAD->MD->getCachedInitialYear();
    MTHREAD->upgradeMainSBLabel("New year started..");
    msgOut(MSG_INFO, "### "+i2s(getYear())+ " year started.. ####");
    time_t now;
    time(&now);
    struct tm *current = localtime(&now);
    string timemessage = "("+i2s(current->tm_hour)+":"+i2s(current->tm_min)+":"+i2s(current->tm_sec)+")";
    MTHREAD->upgradeYearSBLabel(iteration+MTHREAD->MD->getIntSetting("initialYear"));
    MTHREAD->treeViewerChangeGeneralPropertyValue("year", i2s(iteration+ MTHREAD->MD->getIntSetting("initialYear")));
    if(MTHREAD->MD->getBoolSetting("usePixelData")){
      //MTHREAD->GIS->initLayersModelData(); // removed 20120930, not needed, as data in specific pixel values
      MTHREAD->SCORE->runSimulationYear();
    } else {
      MTHREAD->CORE->runSimulationYear();
    }


    //MTHREAD->DO->print(); // done within modelcore now

    for(int i=0;i<MTHREAD->GIS->getXNPixels();i++){
      MTHREAD->GIS->getPixel(i)->newYear(); //delete objects for the pixels, in the update the agents will do the same for their objects
    }
  }
}

```

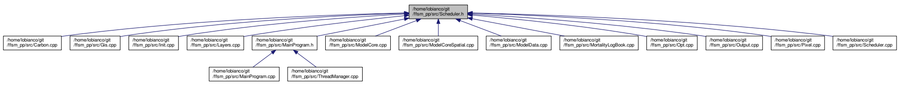
5.133 /home/lobianco/git/ffsm_pp/src/Scheduler.h File Reference

```
#include <string>
#include <vector>
#include <stdexcept>
#include <iostream>
#include <sstream>
#include "BaseClass.h"
```

Include dependency graph for Scheduler.h:



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



Classes

- class Scheduler

  Manage the yearly loops.

5.134 Scheduler.h

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d'Economie Forestière               *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 3 of the License, or     *
 *   (at your option) any later version, given the compliance with the     *
 *   exceptions listed in the file COPYING that is distribued together     *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,       *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of        *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the         *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#ifndef STDSCHEDULER_H
#define STDSCHEDULER_H

#include <string>
#include <vector>
#include <stdexcept>
#include <iostream>
#include <sstream>

// regmas headers..
#include "BaseClass.h"
//#include "ModelData.h"

/// Manage the yearly  loops

/**
This class is responsable to manage the time-dimension of the program.
<br>It start its job when Init has ended and schedule the various operation to be done during the year
 loops.
```

```
@author Antonello Lobianco
*/
class Scheduler: public BaseClass{

public:
                      Scheduler(ThreadManager* MTHREAD_h);
                     ~Scheduler();
  void                run();
  int                 getIteration(){return iteration;};
  int                 getYear(){return year;}
  int                 setYear(const int& year_h){year = year_h;}
  int                 advanceYear(){year += 1;}

private:
  int                                iteration;
  int                                     year;

};

#endif
```

5.135 /home/lobianco/git/ffsm_pp/src/src.pro File Reference

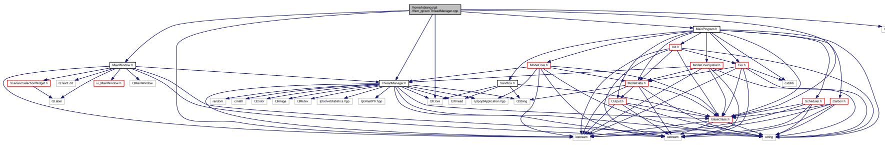
5.136 src.pro

```
########################################################################
# Project file for the FFSM Forest Model
# http://www.ffsm-model.org
#
# You need the Qt GUI framework to use this file.
########################################################################

QT += xml
QT += widgets
DESTDIR = ..
#TARGET = ffsm

unix {
     #LIBS += -lipopt
     LIBS += -ladolc
     LIBS += -lz # needed in Qt5/ ubuntu 13.10 64bit
     #LIBS += -lColPack
     INCLUDEPATH += /usr/include/coin
     INCLUDEPATH += /usr/include/coin/ThirdParty
     INCLUDEPATH += /usr/include/adolc
     INCLUDEPATH += `PKG_CONFIG_PATH=/usr/lib/pkgconfig:/usr/share/pkgconfig: /usr/bin/pkg-config --cflags ipopt` $(ADDINCFLAGS)
     LIBS += `PKG_CONFIG_PATH=/usr/lib/pkgconfig:/usr/share/pkgconfig: /usr/bin/pkg-config --libs ipopt`
     # Next line if we want compile also Coin::Flop++ models:
     # LIBS += `PKG_CONFIG_PATH=/usr/lib64/pkgconfig:/usr/lib/pkgconfig:/usr/share/pkgconfig: pkg-config --libs flopcpp osi-cbc osi-clp`
     #LIBS += -lcoinmetis -lcoinmumps -lblas -llapack
     #LIBS += -lpthread -lgfortran -lcoinmetis -lblas
     #QMAKE_CXXFLAGS_RELEASE += -O3 -pipe -DNDEBUG -pedantic-errors -Wparentheses -Wreturn-type -Wcast-qual -Wall -Wpointer-arith -Wwrite-strings -Wconversion -Wno-unknown-pragmas -Wno-long-long   -DIPOPT_BUILD
     #QMAKE_LFLAGS_RELEASE +=  -Wl,--rpath -Wl,/usr/lib
     #QMAKE_LFLAGS += -lz
     #QMAKE_LFLAGS_RELEASE += -lz
     #QMAKE_LFLAGS_DEBUG += -lz


}

win32 {

    INCLUDEPATH += ThirdParty/win32/include/coin
    INCLUDEPATH += ThirdParty/win32/include/coin/ThirdParty
    INCLUDEPATH += ThirdParty/win32/include
    INCLUDEPATH += ThirdParty/win32/include/adolc
    INCLUDEPATH += $$[QT_INSTALL_DATA]/src/3rdparty/zlib
    LIBS += -L ThirdParty/win32/lib -lipopt
    LIBS += -L ThirdParty/win32/lib -lcoinmetis
    LIBS += -L ThirdParty/win32/lib -lcoinmumps
    LIBS += -L ThirdParty/win32/lib -lcoinhsl
    LIBS += -L ThirdParty/win32/lib -lcoinblas
    LIBS += -L ThirdParty/win32/lib -lcoinlapack
    LIBS += -L ThirdParty/win32/lib -ladolc
    LIBS += -lpthread -lgfortran -lcoinmetis -lcoinblas
    #CONFIG +=  console
    CONFIG += exceptions
}

INCLUDEPATH += ThirdParty/allos/include


TEMPLATE = app
DEPENDPATH += ". agents"
CONFIG += warn_on \
          qt \
      thread \
      debug_and_release
#CONFIG -= release

QMAKE_CXXFLAGS += -std=c++0x

#QMAKE_CXXFLAGS_RELEASE -= -O2
#QMAKE_CXXFLAGS_RELEASE += -O3

#QMAKE_LFLAGS_RELEASE -= -O1

# testing..
#CONFIG += link_pkgconfig
#PKGCONFIG += ipopt
#PKGCONFIG += coinasl
#PKGCONFIG += coinmetis
#PKGCONFIG += ipoptamplinterface
#PKGCONFIG += coinmumps



#INCLUDEPATH += ". agents"


#OBJECTS_DIR = ../bin


CONFIG(release, debug|release) {
    TARGET = ffsm
}
CONFIG(debug, debug|release) {
    TARGET = ffsm_debug
}

#Release:DESTDIR = ../build/release
#Release:TARGET = ffsm
Release:OBJECTS_DIR = ../build/release
Release:MOC_DIR = ../build/release
Release:RCC_DIR = ../build/release
#Release:UI_DIR = ../build/release # then th header file can't find the other headers!

#Debug:DESTDIR = ../build/debug
#Debug:TARGET = ffsm_debug
Debug:OBJECTS_DIR = ../build/debug
Debug:MOC_DIR = ../build/debug
Debug:RCC_DIR = ../build/debug
#Debug:UI_DIR = ../build/debug


# Input
HEADERS += Adolc_debugtest.h \
           CommonLib.h \
           BaseClass.h \
           Gis.h \
           Init.h \
           InputNode.h \
           Ipopt_nlp_problem_debugtest.h \
           MainProgram.h \
           MainWindow.h \
           ModelData.h \
           ModelRegion.h \
           #Set.h \
           Opt.h \
           Output.h \
           Pixel.h \
           Sandbox.h \
           Scheduler.h \
           ThreadManager.h \
           MapBox.h  \
           Layers.h \
           unzip.h \
           unzip_p.h \
           zip.h \
           zip_p.h \
           zipentry_p.h \
           anyoption.h \
           ScenarioSelectionWidget.h \
           ModelCore.h \
           ModelCoreSpatial.h \
           Carbon.h

FORMS    += MainWindow.ui
SOURCES += Adolc_debugtest.cpp \
           CommonLib.cpp \
           BaseClass.cpp \
           Gis.cpp \
           Init.cpp \
           Ipopt_nlp_problem_debugtest.cpp\
           InputNode.cpp \
           main.cpp \
           MainProgram.cpp \
           ModelData.cpp \
           ModelRegion.cpp \
           #Set.cpp \
           Opt.cpp \
           Output.cpp \
           Pixel.cpp \
           Scheduler.cpp \
           Sandbox.cpp \
           ThreadManager.cpp \
           MainWindow.cpp \
           MapBox.cpp \
           Layers.cpp \
           unzip.cpp \
           zip.cpp \
           anyoption.cpp \
           ScenarioSelectionWidget.cpp \
           ModelCore.cpp \
           ModelCoreSpatial.cpp \
           Carbon.cpp

RESOURCES += resources.qrc

```

5.137 /home/lobianco/git/ffsm_pp/src/ThreadManager.cpp File Reference

```
#include <iostream>
#include <QtCore>
#include <QMutexLocker>
#include "ThreadManager.h"
#include "BaseClass.h"
#include "MainProgram.h"
#include "MainWindow.h"
```

Include dependency graph for ThreadManager.cpp:



5.138 ThreadManager.cpp

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière              *
 *   http://ffsm-project.org                                              *
 *                                                                        *
 *   This program is free software; you can redistribute it and/or modify *
 *   it under the terms of the GNU General Public License as published by *
 *   the Free Software Foundation; either version 3 of the License, or    *
 *   (at your option) any later version, given the compliance with the    *
 *   exceptions listed in the file COPYING that is distribued together    *
 *   with this file.                                                      *
 *                                                                        *
 *   This program is distributed in the hope that it will be useful,      *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of       *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the        *
 *   GNU General Public License for more details.                         *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License     *
 *   along with this program; if not, write to the                         *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#include <iostream>

#include <QtCore>
//#include <QMutex>
#include <QMutexLocker>

#include "ThreadManager.h"
#include "BaseClass.h"
#include "MainProgram.h"
#include "MainWindow.h"

using namespace std;

ThreadManager::ThreadManager(){
  running=false;
  stopped=false;
  layerQueryPos = -1;

  // initializing pointers...
  MD      = NULL;
  GIS     = NULL;
  INIT    = NULL;
  SCD     = NULL;
  DO      = NULL;
  CORE    = NULL;
  SCORE   = NULL;
  TEST    = NULL;
  CBAL    = NULL;
  //randev = NULL;
  gen     = NULL;

  GUI = false;

  scenarioName="";
  inputFileName="";
  baseDirectory="";

}

void
ThreadManager::setMessage(const QString &message){
    messageStr = message;
}

void ThreadManager::run(){
  running=true;
  stopped=false;

  srand(1);
  GUI=true;

  emit upgradeLogArea("**INFO: Start running the model...");

  MainProgram* myProgram;
  try{
    deleteDeadOldPointers();
    emit resetGUIForNewSimulation();


    QFileInfo file(inputFileName);
    QDir baseDir = file.absoluteDir();
    baseDirectory = baseDir.absolutePath()+"/";
    myProgram = new MainProgram(this);

    //myProgram->setBaseDirectory(baseDirectory);

    vector<string> scenarios = MD->getScenarios();
    QVector<QString> qscenarios;
    for(uint i=0;i<scenarios.size();i++){
      qscenarios.push_back(scenarios.at(i).c_str());
    }
    running = false;
    emit sendScenarioOptionsToGUI(qscenarios);
    refreshGUI();

    myProgram->run();

    // Here the model has come to an end...
    running=false;
    stopped=true;
    delete myProgram;
```

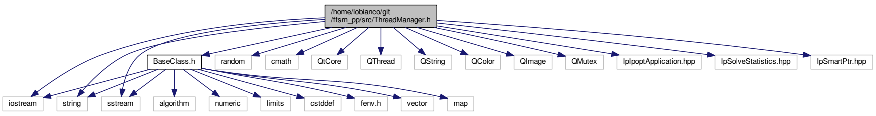
```cpp
    refreshGUI();

  }catch (...) { /// \todo .. perform a better exception handing..
    emit upgradeLogArea("**INFO: Model has stopped or rised an error (read previous line).");
  }
  emit upgradeLogArea("**INFO: Model has ended.");

}

void
ThreadManager::retrieveScenarioNameFromGUI(const QString & scenarioName_h){
  scenarioName = scenarioName_h;
  msgOut(MSG_INFO, "Selected scenario: "+scenarioName.toStdString());
  cout << "Selected scenario: "+scenarioName.toStdString() << endl;
  resume();
}

void
ThreadManager::runFromConsole(QString inputFileName_h, QString scenarioName_h) {
    GUI = false;
    scenarioName = scenarioName_h;
    inputFileName = inputFileName_h;
    QFileInfo file(inputFileName);
    QDir baseDir = file.absoluteDir();
    baseDirectory = baseDir.absolutePath()+"/";
    cout <<"Using base directory: "<< baseDirectory.toStdString() << endl;


    MainProgram* myProgram = new MainProgram(this);

    if( scenarioName_h == ""){ // if the scenario option has not been choosed, go for the first one!
      vector<string> scenarios = MD->getScenarios();
      scenarioName = scenarios.at(0).c_str();
    }

    //myProgram->setBaseDirectory(baseDirectory);
    myProgram->run();
}

void
ThreadManager::setInputFileName(QString inputFileName_h){
  inputFileName= inputFileName_h;
  QFileInfo file(inputFileName);
  QDir baseDir = file.absoluteDir();
  baseDirectory = baseDir.absolutePath()+"/";
}

/**
Delete the pointers (e.g. GIS) eventually remained from a previous run.
<br>This function is called at the START of a new simulation, and it will check if model pointers (e.g. GIS) exist , and if so it will delete them.
<br>This is useful when we keep the MainWindow open but we run the model for a second time.
<br>Why we don't delete them at the end of a simulation, instead of deleting them on a new run? That's because we want let the user to interface with the model even when this is ended, w.g. for query the map.
*/
void
ThreadManager::deleteDeadOldPointers(){
  if (DO) {delete DO; DO=0;}
  if (INIT) {delete INIT; INIT=0;}
  if (SCD) {delete SCD; SCD=0;}
  if (GIS) {delete GIS; GIS=0;}
  if (MD) {delete MD; MD=0;}
  if (CORE){delete CORE; CORE=0;}
  if (SCORE){delete SCORE; SCORE=0;}
  if (CBAL) {delete CBAL; CBAL=0;}
  //if (OPT) {delete OPT; OPT=0;} // not needed, it's a "smart point"
  if(TEST){delete TEST; TEST=0;}
  //if(randev){delete randev; randev=0;}
  if(gen){delete gen; gen=0;}
}

void
ThreadManager::stop(){
    stopped = true;
  emit upgradeLogArea("STOP cliccked stopping");
}

void
ThreadManager::pauseOrResume(){
  if(!stopped){
    if(running){
      running= false;
      emit upgradeLogArea("PAUSE cliccked PAUSING");
    }
    else {
      running=true;
      emit upgradeLogArea("PAUSE cliccked RESUMING");
      emit setGUIUnsavedStatus(true);
    }
  }
  return;
}

void
ThreadManager::pause(){
  if(!stopped){
    if(running){
      running= false;
    }
    else {
      return;
    }
  }
  return;
}

void
ThreadManager::resume(){
  if(!stopped){
    if(running){
      return;
    }
    else {
      running=true;
      emit setGUIUnsavedStatus(true);
    }
  }
  return;
}

void
ThreadManager::refreshGUI(){
    checkQuery(0,0,false);
    while (!running){
      if(stopped){
        break;
      }
    }
    if (stopped){
      emit upgradeLogArea("Model has been stopped.");
      running= false;
      throw(2);
    }
}

void
ThreadManager::msgOut(const int msgCode_h, const string message_h){
  QString message = message_h.c_str();
  emit upgradeLogArea(message);
  if (msgCode_h == 2){
    emit upgradeMainSBLabelToGui(message);
  }
}

void
ThreadManager::setOutputDirName(string outputDirname_h){
  emit setOutputDirNameToGui(outputDirname_h);
}

void
ThreadManager::addLayer(string layerName_h, string layerLabel_h){
  QString layerName = layerName_h.c_str();
  QString layerLabel = layerLabel_h.c_str();
  emit addLayerToGui(layerName, layerLabel);
}

void
ThreadManager::updatePixel(string layerName_h, int x_h, int y_h, QColor color_h){
  emit updatePixelToGui(layerName_h.c_str(), x_h, y_h, color_h);
}

void
ThreadManager::updateImage(string layerName_h, const QImage &image_h){
  emit updateImageToGui(layerName_h.c_str(), image_h);
}

void
ThreadManager::upgradeMainSBLabel(const string message_h){
  emit upgradeMainSBLabelToGui(message_h.c_str());
}

void
```

```
ThreadManager::upgradeYearSBLabel(int year){
  QString temp;
  temp= i2s(year).c_str();
  emit upgradeYearSBLabelToGui(temp);
}

/**
checkQuery() is a function that can be called my the GUI trough a signal or from the running thread under refreshGUI(), and it is protected with a mutex.
<br>It's role is to control the status of pxQueryID and layerQueryPos member variables.
<br>If the call come from the GUI, it is a new request and we set them to the new values, otherwise we gonna see if they are just beed changed and if so (layerQueryPos>=0) we call computeQuery().
*/
void
ThreadManager::checkQuery(int px_ID, int currentLayerIndex, bool newRequest){
  QMutexLocker locker(&mutex);
  if(newRequest){
    pxQueryID = px_ID;
    layerQueryPos = currentLayerIndex;
    if(stopped){computeQuery(pxQueryID, layerQueryPos);layerQueryPos = -1;} // model is stopped, no way the model thread will do the query work
    else{emit publishQueryResults("<i>..wait.. processing query..</i>");} // model is running.. it will be the model thread to execute the query
    return;
  } else {
    if(layerQueryPos<0){
      return;
    } else {
      computeQuery(pxQueryID, layerQueryPos);
      layerQueryPos = -1;
      return;
    }
  }
}

void
ThreadManager::computeQuery(int px_ID, int currentLayerIndex){

  // IMPORTANT: this function is called at refreshGUI() times, so if there are output messages, call them with the option to NOT refresh the gui, otherwise we go to an infinite loop...

  vector<Layers*> layers;
  try {
    layers = GIS->getLayerPointers();
  }catch (...) {
    emit activateTab(2); // tell the gui to activate the 3rd page, those with the pixel info
    emit publishQueryResults("GIS pointer is dead.. maybe simulation has ended???");
    return;
  }
  QString result= "";
  int realID = GIS->sub2realID(px_ID);
  if (realID<0) {
    emit publishQueryResults("Query result: Spatial data is not yet ready in the model. Please click again later.");
    return; // on early stage we may have errors, and here we prevent this error to have further consequences.
  }
  Pixel* px;
  try {
    px = GIS->getPixel(realID);
  }catch (...) {
    emit activateTab(2); // tell the gui to activate the 3rd page, those with the pixel info
    emit publishQueryResults("Query result: Spatial data is not yet ready in the model. Please click again later.");
    return;
  }
  result += "Pixel: ";
  result += i2s(realID).c_str();
  result += " (";
  result += i2s(px->getX()).c_str();
  result += ",";
  result += i2s(px->getY()).c_str();
  result += ")";
  result +="<p><table>";
  uint countVisibleLayers = 0;
  for (uint i=0;i<layers.size();i++){
    if(!layers[i]->getDisplay()){
      continue;
    }
    QString boldStart="";
    QString boldEnd = "";
    if (countVisibleLayers == currentLayerIndex){
      boldStart = "<b>";
      boldEnd   = "</b>";
    }
    result += "<tr>";
    string layerName = layers[i]->getName();
    double value = px->getDoubleValue(layerName);
    string category = layers[i]->getCategory(value);
    //QColor color = layers[i]->getColor(value);
    result += "<td>";
    result += boldStart;
    result += layerName.c_str();
    result += boldEnd;
    result += "</td><td>";
    result += boldStart;
    result += category.c_str();
    result += boldEnd;
    result += "</td>";
    result += "</tr>";
    if(layers[i]->getDisplay()){ // if not really needed, but ok if we decide to change and get displayed also hidden layers
      countVisibleLayers++;
    }
  }
  result += "</table>";
  emit activateTab(2); // tell the gui to activate the 3rd page, those with the pixel info
  emit publishQueryResults(result);
}


```

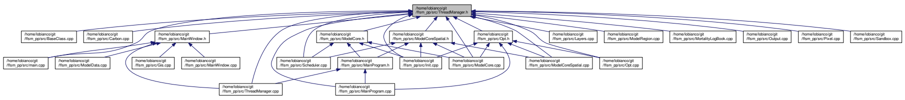
5.139 /home/lobianco/git/ffsm_pp/src/ThreadManager.h File Reference

```
#include <iostream>
#include <string>
#include <sstream>
#include <random>
#include <cmath>
#include <QtCore>
#include <QThread>
#include <QString>
#include <QColor>
#include <QImage>
#include <QMutex>
#include "IpIpoptApplication.hpp"
#include "IpSolveStatistics.hpp"
#include "IpSmartPtr.hpp"
#include "BaseClass.h"
```

Include dependency graph for ThreadManager.h:



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



Classes

- class ThreadManager

  Thread manager. Responsable to manage the main thread and "speak" with the GUI.

5.140 ThreadManager.h

```
/***************************************************************************
 *   Copyright (C) 2015 by Laboratoire d’Economie Forestière             *
 *   http://ffsm-project.org                                               *
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify *
 *   it under the terms of the GNU General Public License as published by *
 *   the Free Software Foundation; either version 3 of the License, or    *
 *   (at your option) any later version, given the compliance with the    *
 *   exceptions listed in the file COPYING that is distribued together    *
 *   with this file.                                                       *
 *                                                                         *
 *   This program is distributed in the hope that it will be useful,      *
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of       *
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the        *
 *   GNU General Public License for more details.                          *
 *                                                                         *
 *   You should have received a copy of the GNU General Public License    *
 *   along with this program; if not, write to the                        *
 *   Free Software Foundation, Inc.,                                       *
 *   59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.             *
 ***************************************************************************/
#ifndef THREAD_H
#define THREAD_H

#include <iostream>
#include <string>
#include <sstream>
#include <random>
#include <cmath>

#include <QtCore>
#include <QThread>
#include <QString>
#include <QColor>
#include <QImage>
#include <QMutex>

#include "IpIpoptApplication.hpp"
#include "IpSolveStatistics.hpp"
#include "IpSmartPtr.hpp"

// regmas includes..
#include "BaseClass.h"

class MainWindow;
class ModelData;
class Gis;
class Init;
class Scheduler;
class Output;
class ModelCore;
class ModelCoreSpatial;
class Opt;
class Sandbox;
class Carbon;

using namespace std;

/// Thread manager. Responsable to manage the main thread and "speak" with the GUI

/**
ThreadManager is responsable for the actions on the main thread (run/pause/resume/stop) and to speack with the GUI using the signal/slot tecniques.
@author Antonello Lobianco
*/
class ThreadManager : public QThread, public BaseClass
{
    Q_OBJECT

public:
                      ThreadManager();
  // pointers..
  ModelData*          MD;    ///< the model data object
  Gis*                GIS;   ///< GIS information and methods
  Init*               INIT;  ///< the Init object (pre-simulation scheduler)
  Scheduler*          SCD;   ///< the scheduler object (simulation-loops scheduler)
  Output*             DO;    ///< data output
  ModelCore*          CORE;  ///< Core of the model
  ModelCoreSpatial*   SCORE; ///< Core of the model (spatial version)
  Carbon*             CBAL;  ///< Module for the Carbon Balance
  Sandbox*            TEST;  ///< Various debugging code for development
  Ipopt::SmartPtr <Ipopt::TNLP>      OPT;  ///< Market optimisation
  //std::random_device*  randev; ///< used in the sampling from normal distribution 20150928: all random_device has been just be replaced with mt19937(time(0)), as largelly enought!
  std::mt19937*        gen;  ///< used in the sampling from normal distribution

  void                 setMessage(const QString &message);
  void                 stop();
  void                 deleteDeadOldPointers(); ///< Useful for several model running without leaving the GUI
  void                 pauseOrResume();
  void                 pause();
  void                 resume();
  void                 refreshGUI();
  void                 msgOut(const int msgCode_h, const string message_h);
  void                 addLayer(string layerName_h, string layerLabel_h);
  void                 updatePixel(string layerName_h, int x_h, int y_h, QColor color);
  void                 updateImage(string layerName_h, const QImage &image_h);
  void                 upgradeMainSBLabel(const string message_h);
  void                 upgradeYearSBLabel(int year);
  string               getBaseDirectory(){return baseDirectory.toStdString();};
  string               getInputFileName(){return inputFileName.toStdString();};
  string               getScenarioName() {return scenarioName.toStdString();};
  void                 setScenarioName(const string &scenarioName_h){scenarioName= scenarioName_h.c_str();};
  void                 setOutputDirName(string outputDirname_h);

  /// the regional data object..
  void                 setMDPointer(ModelData *MD_h){MD=MD_h;};
  /// GIS information and methods..
  void                 setGISPointer(Gis *GIS_h){GIS=GIS_h;};
  /// the Init object, it schedule the pre-simulation phase..
  void                 setINITPointer(Init *INIT_h){INIT=INIT_h;};
  /// the sandbox object for within-development quick tests
  void                 setTestPointer(Sandbox *TEST_h){TEST=TEST_h;};
  /// the scheduler object. It manage the simulation loops..
  void                 setSCDPointer(Scheduler *SCD_h){SCD=SCD_h;};
  /// manage the printing of data needed for scenario-analisys. The "message output" (needed to see "what is it happening?" are instead simply printed with msgOut()..
  void                 setDOPointer(Output *DO_h){DO=DO_h;};
  /// Perform the algorithms of the model
  void                 setCOREPointer(ModelCore* CORE_h){CORE=CORE_h;};
  /// Perform the algorithms of the model
  void                 setSCOREPointer(ModelCoreSpatial* SCORE_h){SCORE= SCORE_h;};
  /// Perform the market optimisation
  void                 setOPTPointer(Ipopt::SmartPtr<Ipopt::TNLP> OPT_h){OPT=OPT_h;};
  /// Module that account for the Carbon Balance
  void                 setCBALPointer(Carbon *CBAL_h){CBAL=CBAL_h;};

  //public slots:
  void                 setInputFileName(QString inputFileName_h);
  //void                 setBaseDirectory(QString baseDirectory_h){baseDirectory =  baseDirectory_h;};

  // tree viewer operations...
  /*
  void                treeViewerAddManager(string name){emit treeViewerAddItemToGui("Manager "+name, "manager_"+name, "managers");};
  void                treeViewerAddAgent(int uniqueID){emit treeViewerAddItemToGui(i2s(uniqueID), "agent_"+i2s(uniqueID), "agents");  };
  void                treeViewerAddManagerProperty(string managerName, string propertyName){
                          emit treeViewerAddItemToGui(propertyName, "manager_"+managerName+"_"+propertyName, "manager_"+managerName);};
  void              treeViewerAddAgentProperty(int uniqueID, string propertyName){
                          emit treeViewerAddItemToGui(propertyName, "agent_"+i2s(uniqueID)+"_"+propertyName, "agent_"+i2s(uniqueID));};
  void                treeViewerManagerPropertyChangeValue(string managerName, string propertyName, string newValue){
                          emit treeViewerItemChangeValueToGui("manager_"+managerName+"_"+propertyName, newValue);};
  void                treeViewerAgentPropertyChangeValue(int uniqueID, string propertyName, string newValue){
                          emit treeViewerItemChangeValueToGui("agent_"+i2s(uniqueID)+"_"+propertyName, newValue);};
  void                treeViewerRemoveManager(string name){emit treeViewerItemRemoveToGui("manager_"+name);};
  void                treeViewerRemoveAgent(int uniqueID){emit treeViewerItemRemoveToGui("agent_"+i2s(uniqueID));};
  */
  void                treeViewerChangeGeneralPropertyValue(string propertyName, string newValue){
                          emit treeViewerItemChangeValueToGui("general_"+propertyName, newValue);};


  void                 fitInWindow() {emit fitInWindowToGui();}; ///< Re-draw the map making it to fit (with the right proportions) to the widget
  void                 runFromConsole(QString inputFileName_h, QString scenarioName_h);
  bool                 usingGUI(){return GUI;};

signals:
  void                 upgradeLogArea(const QString &logMessage);
```

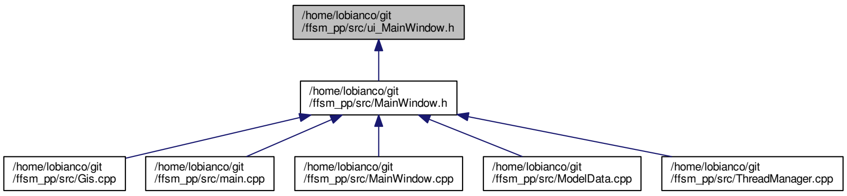
```
  void                upgradeMainSBLabelToGui(const QString &logMessage);
  void                upgradeYearSBLabelToGui(const QString &logMessage);
  void                addLayerToGui(QString layerName, QString layerLabel);
  void                updatePixelToGui(QString layerName_h, int x_h, int y_h, QColor color);
  void                updateImageToGui(QString layerName_h, QImage image_h);
  void                setOutputDirNameToGui(string outputDirname_h);
  void                setGUIUnsavedStatus(bool status_h);
  void                setGUIMapDimension(int x_h, int y_h);
  void                treeViewerItemChangeValueToGui(string itemID, string newValue);
  void                treeViewerItemRemoveToGui(string itemID);
  void                treeViewerAddItemToGui(string text, string itemID, string parentID);
  void                fitInWindowToGui();
  void                queryRequestOnPx(int px_ID, int currentLayerIndex);
  void                publishQueryResults(const QString &results);
  void                activateTab(int pos_h);
  void                resetGUIForNewSimulation();
  void                sendScenarioOptionsToGUI(const QVector<QString> &scenarios_h);


public slots:
  /// Switch and control the access to pxQueryID and layerQueryPos members
  void                checkQuery(int px_ID, int currentLayerIndex, bool newRequest=true);
  /// Compute the pixel query and return it to the GUI (with a signal)
  void                computeQuery(int px_ID, int currentLayerIndex);
  void                retrieveScenarioNameFromGUI(const QString &scenarioName_h);

protected:
  void                run();

private:
  QString                           messageStr;
  volatile bool                        stopped;
  volatile bool                        running;
  QString                        inputFileName;
  QString                        baseDirectory;
  QString                         scenarioName;
  volatile int                       pxQueryID;
  volatile int                   layerQueryPos;
  QMutex                                 mutex;
  bool                                     GUI;

};



#endif

```

5.141 /home/lobianco/git/ffsm_pp/src/ui_MainWindow.h File Reference

```
#include <QtCore/QLocale>
#include <QtCore/QVariant>
#include <QtWidgets/QAction>
#include <QtWidgets/QApplication>
#include <QtWidgets/QButtonGroup>
#include <QtWidgets/QComboBox>
#include <QtWidgets/QGridLayout>
#include <QtWidgets/QHBoxLayout>
#include <QtWidgets/QHeaderView>
#include <QtWidgets/QMainWindow>
#include <QtWidgets/QMenu>
#include <QtWidgets/QMenuBar>
#include <QtWidgets/QPushButton>
#include <QtWidgets/QSpacerItem>
#include <QtWidgets/QSplitter>
#include <QtWidgets/QStatusBar>
#include <QtWidgets/QTabWidget>
#include <QtWidgets/QTextEdit>
#include <QtWidgets/QToolBar>
#include <QtWidgets/QTreeWidget>
#include <QtWidgets/QVBoxLayout>
#include <QtWidgets/QWidget>
#include "MapBox.h"
```

Include dependency graph for ui_MainWindow.h:

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



Classes

- class Ui_MainWindow
- class MainWindow

Namespaces

- Ui

5.142 ui_MainWindow.h

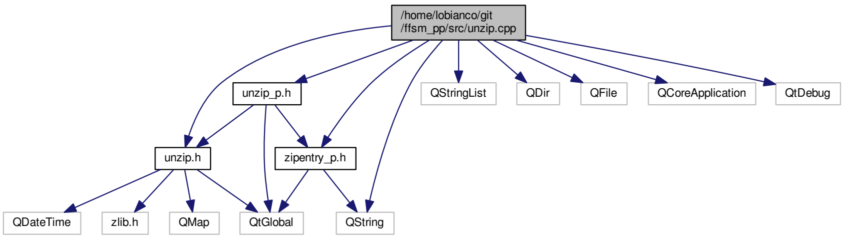
```
/********************************************************************************
** Form generated from reading UI file 'MainWindow.ui'
**
** Created by: Qt User Interface Compiler version 5.5.1
**
** WARNING! All changes made in this file will be lost when recompiling UI file!
********************************************************************************/

#ifndef UI_MAINWINDOW_H
#define UI_MAINWINDOW_H

#include <QtCore/QLocale>
#include <QtCore/QVariant>
#include <QtWidgets/QAction>
#include <QtWidgets/QApplication>
#include <QtWidgets/QButtonGroup>
#include <QtWidgets/QComboBox>
#include <QtWidgets/QGridLayout>
#include <QtWidgets/QHBoxLayout>
#include <QtWidgets/QHeaderView>
#include <QtWidgets/QMainWindow>
#include <QtWidgets/QMenu>
#include <QtWidgets/QMenuBar>
#include <QtWidgets/QPushButton>
#include <QtWidgets/QSpacerItem>
#include <QtWidgets/QSplitter>
#include <QtWidgets/QStatusBar>
#include <QtWidgets/QTabWidget>
#include <QtWidgets/QTextEdit>
#include <QtWidgets/QToolBar>
#include <QtWidgets/QTreeWidget>
#include <QtWidgets/QVBoxLayout>
#include <QtWidgets/QWidget>
#include "MapBox.h"

```

```
QT_BEGIN_NAMESPACE

class Ui_MainWindow
{
public:
    QAction *actionLoadConfiguration;
    QAction *actionSaveLog;
    QAction *actionSaveLogAs;
    QAction *actionRun;
    QAction *actionPause;
    QAction *actionStop;
    QAction *actionAboutRegMAS;
    QAction *actionExit;
    QAction *actionHideDebugMsgs;
    QAction *actionRegMASDocumentation;
    QAction *actionFitMap;
    QAction *actionViewResults;
    QWidget *centralwidget;
    QHBoxLayout *hboxLayout;
    QSplitter *splitter;
    QWidget *layoutWidget;
    QVBoxLayout *vboxLayout;
    QComboBox *layerSelector;
    QSpacerItem *spacerItem;
    MapBox *mapBox;
    QTabWidget *tabWidget;
    QWidget *log_area;
    QVBoxLayout *verticalLayout;
    QTextEdit *logArea;
    QPushButton *viewResultsButton;
    QWidget *model_viewer;
    QHBoxLayout *hboxLayout1;
    QTreeWidget *statusView;
    QWidget *plot_info;
    QGridLayout *gridLayout;
    QTextEdit *pxInfoArea;
    QMenuBar *menubar;
    QMenu *menuView;
    QMenu *menuHelp;
    QMenu *menuAction;
    QMenu *menuFile;
    QStatusBar *statusbar;
    QToolBar *modelToolBar;
    QToolBar *fileToolBar;

    void setupUi(QMainWindow *MainWindow)
    {
        if (MainWindow->objectName().isEmpty())
            MainWindow->setObjectName(QStringLiteral("MainWindow"));
        MainWindow->setWindowModality(Qt::ApplicationModal);
        MainWindow->resize(667, 467);
        QSizePolicy sizePolicy(QSizePolicy::Fixed, QSizePolicy::Fixed);
        sizePolicy.setHorizontalStretch(1);
        sizePolicy.setVerticalStretch(1);
        sizePolicy.setHeightForWidth(MainWindow->sizePolicy().hasHeightForWidth());
        MainWindow->setSizePolicy(sizePolicy);
        QIcon icon;
        icon.addFile(QStringLiteral(":/imgs/icon.png"), QSize(), QIcon::Normal, QIcon::Off);
        MainWindow->setWindowIcon(icon);
        MainWindow->setIconSize(QSize(24, 24));
        actionLoadConfiguration = new QAction(MainWindow);
        actionLoadConfiguration->setObjectName(QStringLiteral("actionLoadConfiguration"));
        QIcon icon1;
        icon1.addFile(QStringLiteral(":/imgs/open.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionLoadConfiguration->setIcon(icon1);
        actionSaveLog = new QAction(MainWindow);
        actionSaveLog->setObjectName(QStringLiteral("actionSaveLog"));
        QIcon icon2;
        icon2.addFile(QStringLiteral(":/imgs/save.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionSaveLog->setIcon(icon2);
        actionSaveLogAs = new QAction(MainWindow);
        actionSaveLogAs->setObjectName(QStringLiteral("actionSaveLogAs"));
        QIcon icon3;
        icon3.addFile(QStringLiteral(":/imgs/saveas.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionSaveLogAs->setIcon(icon3);
        actionRun = new QAction(MainWindow);
        actionRun->setObjectName(QStringLiteral("actionRun"));
        QIcon icon4;
        icon4.addFile(QStringLiteral(":/imgs/play.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionRun->setIcon(icon4);
        actionPause = new QAction(MainWindow);
        actionPause->setObjectName(QStringLiteral("actionPause"));
        QIcon icon5;
        icon5.addFile(QStringLiteral(":/imgs/pause.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionPause->setIcon(icon5);
        actionStop = new QAction(MainWindow);
        actionStop->setObjectName(QStringLiteral("actionStop"));
        QIcon icon6;
        icon6.addFile(QStringLiteral(":/imgs/stop.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionStop->setIcon(icon6);
        actionAboutRegMAS = new QAction(MainWindow);
        actionAboutRegMAS->setObjectName(QStringLiteral("actionAboutRegMAS"));
        QIcon icon7;
        icon7.addFile(QStringLiteral(":/imgs/info.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionAboutRegMAS->setIcon(icon7);
        actionExit = new QAction(MainWindow);
        actionExit->setObjectName(QStringLiteral("actionExit"));
        QIcon icon8;
        icon8.addFile(QStringLiteral(":/imgs/exit.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionExit->setIcon(icon8);
        actionHideDebugMsgs = new QAction(MainWindow);
        actionHideDebugMsgs->setObjectName(QStringLiteral("actionHideDebugMsgs"));
        actionHideDebugMsgs->setCheckable(true);
        QIcon icon9;
        icon9.addFile(QStringLiteral(":/imgs/clear.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionHideDebugMsgs->setIcon(icon9);
        actionRegMASDocumentation = new QAction(MainWindow);
        actionRegMASDocumentation->setObjectName(QStringLiteral("actionRegMASDocumentation"));
        QIcon icon10;
        icon10.addFile(QStringLiteral(":/imgs/help.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionRegMASDocumentation->setIcon(icon10);
        actionFitMap = new QAction(MainWindow);
        actionFitMap->setObjectName(QStringLiteral("actionFitMap"));
        QIcon icon11;
        icon11.addFile(QStringLiteral(":/imgs/view-refresh.png"), QSize(), QIcon::Normal, QIcon::Off);
        actionFitMap->setIcon(icon11);
        actionViewResults = new QAction(MainWindow);
        actionViewResults->setObjectName(QStringLiteral("actionViewResults"));
        centralwidget = new QWidget(MainWindow);
        centralwidget->setObjectName(QStringLiteral("centralwidget"));
        sizePolicy.setHeightForWidth(centralwidget->sizePolicy().hasHeightForWidth());
        centralwidget->setSizePolicy(sizePolicy);
        hboxLayout = new QHBoxLayout(centralwidget);
        hboxLayout->setObjectName(QStringLiteral("hboxLayout"));
        splitter = new QSplitter(centralwidget);
        splitter->setObjectName(QStringLiteral("splitter"));
        splitter->setOrientation(Qt::Horizontal);
        layoutWidget = new QWidget(splitter);
        layoutWidget->setObjectName(QStringLiteral("layoutWidget"));
        vboxLayout = new QVBoxLayout(layoutWidget);
        vboxLayout->setObjectName(QStringLiteral("vboxLayout"));
        vboxLayout->setContentsMargins(0, 0, 0, 0);
        layerSelector = new QComboBox(layoutWidget);
        layerSelector->setObjectName(QStringLiteral("layerSelector"));
        QSizePolicy sizePolicy1(QSizePolicy::Preferred, QSizePolicy::Fixed);
        sizePolicy1.setHorizontalStretch(1);
        sizePolicy1.setVerticalStretch(0);
        sizePolicy1.setHeightForWidth(layerSelector->sizePolicy().hasHeightForWidth());
        layerSelector->setSizePolicy(sizePolicy1);

        vboxLayout->addWidget(layerSelector);

        spacerItem = new QSpacerItem(200, 16, QSizePolicy::Minimum, QSizePolicy::Expanding);

        vboxLayout->addItem(spacerItem);

        mapBox = new MapBox(layoutWidget);
        mapBox->setObjectName(QStringLiteral("mapBox"));
        QSizePolicy sizePolicy2(QSizePolicy::Expanding, QSizePolicy::Expanding);
        sizePolicy2.setHorizontalStretch(2);
        sizePolicy2.setVerticalStretch(2);
        sizePolicy2.setHeightForWidth(mapBox->sizePolicy().hasHeightForWidth());
        mapBox->setSizePolicy(sizePolicy2);
        mapBox->setMinimumSize(QSize(300, 300));

        vboxLayout->addWidget(mapBox);

        splitter->addWidget(layoutWidget);
        tabWidget = new QTabWidget(splitter);
        tabWidget->setObjectName(QStringLiteral("tabWidget"));
        log_area = new QWidget();
        log_area->setObjectName(QStringLiteral("log_area"));
        verticalLayout = new QVBoxLayout(log_area);
        verticalLayout->setObjectName(QStringLiteral("verticalLayout"));
        logArea = new QTextEdit(log_area);
        logArea->setObjectName(QStringLiteral("logArea"));

        verticalLayout->addWidget(logArea);

        viewResultsButton = new QPushButton(log_area);
        viewResultsButton->setObjectName(QStringLiteral("viewResultsButton"));
        viewResultsButton->setLocale(QLocale(QLocale::English, QLocale::UnitedKingdom));

        verticalLayout->addWidget(viewResultsButton);

        tabWidget->addTab(log_area, QString());
        model_viewer = new QWidget();
        model_viewer->setObjectName(QStringLiteral("model_viewer"));
        hboxLayout1 = new QHBoxLayout(model_viewer);
        hboxLayout1->setObjectName(QStringLiteral("hboxLayout1"));
        statusView = new QTreeWidget(model_viewer);
        statusView->setObjectName(QStringLiteral("statusView"));

        hboxLayout1->addWidget(statusView);

        tabWidget->addTab(model_viewer, QString());
        plot_info = new QWidget();
        plot_info->setObjectName(QStringLiteral("plot_info"));
        gridLayout = new QGridLayout(plot_info);
        gridLayout->setObjectName(QStringLiteral("gridLayout"));
        pxInfoArea = new QTextEdit(plot_info);
        pxInfoArea->setObjectName(QStringLiteral("pxInfoArea"));
        pxInfoArea->setOverwriteMode(false);
        pxInfoArea->setTextInteractionFlags(Qt::TextSelectableByKeyboard|Qt::TextSelectableByMouse);

        gridLayout->addWidget(pxInfoArea, 0, 0, 1, 1);

        tabWidget->addTab(plot_info, QString());
        splitter->addWidget(tabWidget);

        hboxLayout->addWidget(splitter);

        MainWindow->setCentralWidget(centralwidget);
        menubar = new QMenuBar(MainWindow);
        menubar->setObjectName(QStringLiteral("menubar"));
        menubar->setGeometry(QRect(0, 0, 667, 25));
        menuView = new QMenu(menubar);
        menuView->setObjectName(QStringLiteral("menuView"));
        menuHelp = new QMenu(menubar);
        menuHelp->setObjectName(QStringLiteral("menuHelp"));
        menuAction = new QMenu(menubar);
        menuAction->setObjectName(QStringLiteral("menuAction"));
        menuFile = new QMenu(menubar);
        menuFile->setObjectName(QStringLiteral("menuFile"));
        MainWindow->setMenuBar(menubar);
        statusbar = new QStatusBar(MainWindow);
        statusbar->setObjectName(QStringLiteral("statusbar"));
        MainWindow->setStatusBar(statusbar);
        modelToolBar = new QToolBar(MainWindow);
        modelToolBar->setObjectName(QStringLiteral("modelToolBar"));
        modelToolBar->setOrientation(Qt::Horizontal);
        MainWindow->addToolBar(Qt::TopToolBarArea, modelToolBar);
        fileToolBar = new QToolBar(MainWindow);
        fileToolBar->setObjectName(QStringLiteral("fileToolBar"));
        fileToolBar->setOrientation(Qt::Horizontal);
        MainWindow->addToolBar(Qt::TopToolBarArea, fileToolBar);

        menubar->addAction(menuFile->menuAction());
        menubar->addAction(menuAction->menuAction());
        menubar->addAction(menuView->menuAction());
        menubar->addAction(menuHelp->menuAction());
        menuView->addAction(actionHideDebugMsgs);
        menuView->addAction(actionFitMap);
        menuHelp->addAction(actionRegMASDocumentation);
        menuHelp->addAction(actionAboutRegMAS);
        menuAction->addAction(actionRun);
        menuAction->addAction(actionPause);
        menuAction->addAction(actionStop);
        menuFile->addAction(actionLoadConfiguration);
        menuFile->addAction(actionSaveLog);
        menuFile->addAction(actionSaveLogAs);
        modelToolBar->addAction(actionRun);
        modelToolBar->addAction(actionPause);
        modelToolBar->addAction(actionStop);
        fileToolBar->addAction(actionLoadConfiguration);
        fileToolBar->addAction(actionSaveLog);
        fileToolBar->addAction(actionExit);

        retranslateUi(MainWindow);

        tabWidget->setCurrentIndex(0);


        QMetaObject::connectSlotsByName(MainWindow);
    } // setupUi

    void retranslateUi(QMainWindow *MainWindow)
    {
        MainWindow->setWindowTitle(QApplication::translate("MainWindow", "FFSM - Forest Sector Simulator",
0));
        actionLoadConfiguration->setText(QApplication::translate("MainWindow", "&Load Configuration", 0));
        actionSaveLog->setText(QApplication::translate("MainWindow", "&Save log", 0));
        actionSaveLogAs->setText(QApplication::translate("MainWindow", "Save log &as..", 0));
        actionRun->setText(QApplication::translate("MainWindow", "&Run", 0));
        actionPause->setText(QApplication::translate("MainWindow", "&Pause / Resume", 0));
        actionStop->setText(QApplication::translate("MainWindow", "&Stop", 0));
        actionAboutRegMAS->setText(QApplication::translate("MainWindow", "&About RegMAS", 0));
        actionExit->setText(QApplication::translate("MainWindow", "&Exit", 0));
        actionHideDebugMsgs->setText(QApplication::translate("MainWindow", "Hide &debug messages", 0));
        actionRegMASDocumentation->setText(QApplication::translate("MainWindow", "RegMAS &documentation", 0));
        actionFitMap->setText(QApplication::translate("MainWindow", "&Fit map in Window", 0));
        actionViewResults->setText(QApplication::translate("MainWindow", "goToResults", 0));
#ifndef QT_NO_WHATSTHIS
        logArea->setWhatsThis(QApplication::translate("MainWindow", "<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:'Sans Serif'; font-size:9pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\">Run-time logs collecting area (can be saved)</p></body></html>", 0));
#endif // QT_NO_WHATSTHIS
#ifndef QT_NO_TOOLTIP
        viewResultsButton->setToolTip(QApplication::translate("MainWindow", "<html><head/><body><p>You will need a recent version of LibreOffice (or OpenOffice) installed on your system to view the results.</p><p>If you don't have it you can download it from <a href=\"http://www.libreoffice.org\"><span style=\" text-decoration: underline; color:#0000ff;\">http://www.libreoffice.org.</span></a></p><p/></body></html>", 0));
#endif // QT_NO_TOOLTIP
        viewResultsButton->setText(QApplication::translate("MainWindow", "Go to results", 0));
        tabWidget->setTabText(tabWidget->indexOf(log_area), QApplication::translate("MainWindow", "Log area", 0));
        QTreeWidgetItem *___qtreewidgetitem = statusView->headerItem();
        ___qtreewidgetitem->setText(0, QApplication::translate("MainWindow", "1", 0));
#ifndef QT_NO_WHATSTHIS
        statusView->setWhatsThis(QApplication::translate("MainWindow", "<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:'Sans Serif'; font-size:9pt; font-weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\">Run-time viewer of important model status variables</p></body></html>", 0));
#endif // QT_NO_WHATSTHIS
        tabWidget->setTabText(tabWidget->indexOf(model_viewer), QApplication::translate("MainWindow", "Model viewer", 0));
        tabWidget->setTabText(tabWidget->indexOf(plot_info), QApplication::translate("MainWindow", "Plot info", 0));
        menuView->setTitle(QApplication::translate("MainWindow", "&View", 0));
        menuHelp->setTitle(QApplication::translate("MainWindow", "&Help", 0));
        menuAction->setTitle(QApplication::translate("MainWindow", "&Action", 0));
        menuFile->setTitle(QApplication::translate("MainWindow", "&File", 0));
    } // retranslateUi

};

namespace Ui {
    class MainWindow: public Ui_MainWindow {};
} // namespace Ui

QT_END_NAMESPACE

#endif // UI_MAINWINDOW_H
```

5.143 /home/lobianco/git/ffsm_pp/src/unzip.cpp File Reference

```
#include "unzip.h"
#include "unzip_p.h"
#include "zipentry_p.h"
#include <QString>
#include <QStringList>
#include <QDir>
#include <QFile>
#include <QCoreApplication>
#include <QtDebug>
```

Include dependency graph for unzip.cpp:



Macros

- #define UNZIP_LOCAL_HEADER_SIZE 26

  Local header size (excluding signature, excluding variable length fields)
- #define UNZIP_CD_ENTRY_SIZE_NS 42

  Central Directory file entry size (excluding signature, excluding variable length fields)
- #define UNZIP_DD_SIZE 12

  Data descriptor size (excluding signature)
- #define UNZIP_EOCD_SIZE 22

  End Of Central Directory size (including signature, excluding variable length fields)
- #define UNZIP_LOCAL_ENC_HEADER_SIZE 12

  Local header entry encryption header size.
- #define UNZIP_CD_OFF_VERSION 0
- #define UNZIP_CD_OFF_GPFLAG 4
- #define UNZIP_CD_OFF_CMETHOD 6
- #define UNZIP_CD_OFF_MODT 8
- #define UNZIP_CD_OFF_MODD 10
- #define UNZIP_CD_OFF_CRC32 12
- #define UNZIP_CD_OFF_CSIZE 16
- #define UNZIP_CD_OFF_USIZE 20
- #define UNZIP_CD_OFF_NAMELEN 24
- #define UNZIP_CD_OFF_XLEN 26
- #define UNZIP_CD_OFF_COMMLEN 28
- #define UNZIP_CD_OFF_LHOFFSET 38
- #define UNZIP_LH_OFF_VERSION 0
- #define UNZIP_LH_OFF_GPFLAG 2
- #define UNZIP_LH_OFF_CMETHOD 4
- #define UNZIP_LH_OFF_MODT 6
- #define UNZIP_LH_OFF_MODD 8
- #define UNZIP_LH_OFF_CRC32 10
- #define UNZIP_LH_OFF_CSIZE 14
- #define UNZIP_LH_OFF_USIZE 18
- #define UNZIP_LH_OFF_NAMELEN 22
- #define UNZIP_LH_OFF_XLEN 24
- #define UNZIP_DD_OFF_CRC32 0
- #define UNZIP_DD_OFF_CSIZE 4
- #define UNZIP_DD_OFF_USIZE 8
- #define UNZIP_EOCD_OFF_ENTRIES 6
- #define UNZIP_EOCD_OFF_CDOFF 12
- #define UNZIP_EOCD_OFF_COMMLEN 16

- #define UNZIP_VERSION 0x1B
- #define UNZIP_VERSION_STRICT 0x14

    Full compatibility granted until this version.
- #define CRC32(c, b) crcTable[((int)c^b) & 0xff] ^ (c >> 8)

    CRC32 routine.
- #define UNZIP_CHECK_FOR_VALID_DATA

    Checks if some file has been already extracted.

5.143.1 Macro Definition Documentation

5.143.1.1 #define CRC32(*c*, *b*) crcTable[((int)c^b) & 0xff] ^ (c >> 8)

CRC32 routine.

Definition at line 136 of file unzip.cpp.

Referenced by UnzipPrivate::updateKeys().

5.143.1.2 #define UNZIP_CD_ENTRY_SIZE_NS 42

Central Directory file entry size (excluding signature, excluding variable length fields)

Definition at line 81 of file unzip.cpp.

Referenced by UnzipPrivate::parseCentralDirectoryRecord().

5.143.1.3 #define UNZIP_CD_OFF_CMETHOD 6

Definition at line 92 of file unzip.cpp.

Referenced by UnzipPrivate::parseCentralDirectoryRecord().

5.143.1.4 #define UNZIP_CD_OFF_COMMLEN 28

Definition at line 100 of file unzip.cpp.

Referenced by UnzipPrivate::parseCentralDirectoryRecord().

5.143.1.5 #define UNZIP_CD_OFF_CRC32 12

Definition at line 95 of file unzip.cpp.

Referenced by UnzipPrivate::parseCentralDirectoryRecord().

5.143.1.6 #define UNZIP_CD_OFF_CSIZE 16

Definition at line 96 of file unzip.cpp.

Referenced by UnzipPrivate::parseCentralDirectoryRecord().

5.143.1.7 #define UNZIP_CD_OFF_GPFLAG 4

Definition at line 91 of file unzip.cpp.

Referenced by UnzipPrivate::parseCentralDirectoryRecord().

5.143.1.8 #define UNZIP_CD_OFF_LHOFFSET 38

Definition at line 101 of file unzip.cpp.

Referenced by UnzipPrivate::parseCentralDirectoryRecord().

5.143.1.9 #define UNZIP_CD_OFF_MODD 10

Definition at line 94 of file unzip.cpp.

Referenced by UnzipPrivate::parseCentralDirectoryRecord().

5.143.1.10 #define UNZIP_CD_OFF_MODT 8

Definition at line 93 of file unzip.cpp.

Referenced by UnzipPrivate::parseCentralDirectoryRecord().

5.143.1.11 #define UNZIP_CD_OFF_NAMELEN 24

Definition at line 98 of file unzip.cpp.

Referenced by UnzipPrivate::parseCentralDirectoryRecord().

5.143.1.12 #define UNZIP_CD_OFF_USIZE 20

Definition at line 97 of file unzip.cpp.

Referenced by UnzipPrivate::parseCentralDirectoryRecord().

5.143.1.13 #define UNZIP_CD_OFF_VERSION 0

Definition at line 90 of file unzip.cpp.

Referenced by UnzipPrivate::parseCentralDirectoryRecord().

5.143.1.14 #define UNZIP_CD_OFF_XLEN 26

Definition at line 99 of file unzip.cpp.

Referenced by UnzipPrivate::parseCentralDirectoryRecord().

5.143.1.15 #define UNZIP_CHECK_FOR_VALID_DATA

Value:

```
{\
    if (headers != 0)\
    {\
      qDebug() << "Corrupted zip archive. Some files might be extracted.";\
      ec = headers->size() != 0 ? UnZip::PartiallyCorrupted :
      UnZip::Corrupted;\
      break;\
    }\
    else\
    {\
      delete device;\
      device = 0;\
      qDebug() << "Corrupted or invalid zip archive";\
      ec = UnZip::Corrupted;\
      break;\
    }\
  }
```

Checks if some file has been already extracted.

Definition at line 139 of file unzip.cpp.

Referenced by UnzipPrivate::openArchive().

5.143.1.16 #define UNZIP_DD_OFF_CRC32 0

Definition at line 116 of file unzip.cpp.

Referenced by UnzipPrivate::parseLocalHeaderRecord().

5.143.1.17 #define UNZIP_DD_OFF_CSIZE 4

Definition at line 117 of file unzip.cpp.

Referenced by UnzipPrivate::parseLocalHeaderRecord().

5.143.1.18 #define UNZIP_DD_OFF_USIZE 8

Definition at line 118 of file unzip.cpp.

Referenced by UnzipPrivate::parseLocalHeaderRecord().

5.143.1.19 #define UNZIP_DD_SIZE 12

Data descriptor size (excluding signature)

Definition at line 83 of file unzip.cpp.

Referenced by UnzipPrivate::parseLocalHeaderRecord().

5.143.1.20 #define UNZIP_EOCD_OFF_CDOFF 12

Definition at line 122 of file unzip.cpp.

Referenced by UnzipPrivate::seekToCentralDirectory().

5.143.1.21 #define UNZIP_EOCD_OFF_COMMLEN 16

Definition at line 123 of file unzip.cpp.

Referenced by UnzipPrivate::seekToCentralDirectory().

5.143.1.22 #define UNZIP_EOCD_OFF_ENTRIES 6

Definition at line 121 of file unzip.cpp.

Referenced by UnzipPrivate::seekToCentralDirectory().

5.143.1.23 #define UNZIP_EOCD_SIZE 22

End Of Central Directory size (including signature, excluding variable length fields)

Definition at line 85 of file unzip.cpp.

Referenced by UnzipPrivate::seekToCentralDirectory().

5.143.1.24 #define UNZIP_LH_OFF_CMETHOD 4

Definition at line 106 of file unzip.cpp.

Referenced by UnzipPrivate::parseLocalHeaderRecord().

5.143.1.25 #define UNZIP_LH_OFF_CRC32 10

Definition at line 109 of file unzip.cpp.

Referenced by UnzipPrivate::parseLocalHeaderRecord().

5.143.1.26 #define UNZIP_LH_OFF_CSIZE 14

Definition at line 110 of file unzip.cpp.

Referenced by UnzipPrivate::parseLocalHeaderRecord().

5.143.1.27 #define UNZIP_LH_OFF_GPFLAG 2

Definition at line 105 of file unzip.cpp.

Referenced by UnzipPrivate::parseLocalHeaderRecord().

5.143.1.28 #define UNZIP_LH_OFF_MODD 8

Definition at line 108 of file unzip.cpp.

Referenced by UnzipPrivate::parseLocalHeaderRecord().

5.143.1.29 #define UNZIP_LH_OFF_MODT 6

Definition at line 107 of file unzip.cpp.

Referenced by UnzipPrivate::parseLocalHeaderRecord().

5.143.1.30 #define UNZIP_LH_OFF_NAMELEN 22

Definition at line 112 of file unzip.cpp.

Referenced by UnzipPrivate::parseLocalHeaderRecord().

5.143.1.31 #define UNZIP_LH_OFF_USIZE 18

Definition at line 111 of file unzip.cpp.

Referenced by UnzipPrivate::parseLocalHeaderRecord().

5.143.1.32 #define UNZIP_LH_OFF_VERSION 0

Definition at line 104 of file unzip.cpp.

5.143.1.33 #define UNZIP_LH_OFF_XLEN 24

Definition at line 113 of file unzip.cpp.

Referenced by UnzipPrivate::parseLocalHeaderRecord().

5.143.1.34 #define UNZIP_LOCAL_ENC_HEADER_SIZE 12

Local header entry encryption header size.

Definition at line 87 of file unzip.cpp.

Referenced by UnzipPrivate::extractFile().

5.143.1.35 #define UNZIP_LOCAL_HEADER_SIZE 26

Local header size (excluding signature, excluding variable length fields)

Definition at line 79 of file unzip.cpp.

Referenced by UnzipPrivate::parseLocalHeaderRecord().

5.143.1.36 #define UNZIP_VERSION 0x1B

Max version handled by this API. 0x1B = 2.7 –> full compatibility only up to version 2.0 (0x14) versions from 2.1 to 2.7 may use unsupported compression methods versions after 2.7 may have an incompatible header format

Definition at line 131 of file unzip.cpp.

Referenced by UnzipPrivate::parseCentralDirectoryRecord().

5.143.1.37 #define UNZIP_VERSION_STRICT 0x14

Full compatibility granted until this version.

Definition at line 133 of file unzip.cpp.

5.144 unzip.cpp

```
/****************************************************************************
** Filename: unzip.cpp
** Last updated [dd/mm/yyyy]: 28/01/2007
**
** pkzip 2.0 decompression.
**
** Some of the code has been inspired by other open source projects,
** (mainly Info-Zip and Gilles Vollant’s minizip).
** Compression and decompression actually uses the zlib library.
**
** Copyright (C) 2007 Angius Fabrizio. All rights reserved.
**
** This file is part of the OSDaB project (http://osdab.sourceforge.net/).
**
** This file may be distributed and/or modified under the terms of the
** GNU General Public License version 2 as published by the Free Software
** Foundation and appearing in the file LICENSE.GPL included in the
** packaging of this file.
**
** This file is provided AS IS with NO WARRANTY OF ANY KIND, INCLUDING THE
** WARRANTY OF DESIGN, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
**
** See the file LICENSE.GPL that came with this software distribution or
** visit http://www.gnu.org/copyleft/gpl.html for GPL licensing information.
**
**********************************************************************/

#include "unzip.h"
#include "unzip_p.h"
#include "zipentry_p.h"

#include <QString>
#include <QStringList>
#include <QDir>
#include <QFile>
#include <QCoreApplication>

// You can remove this #include if you replace the qDebug() statements.
#include <QtDebug>

/*!
  \class UnZip unzip.h

  \brief PKZip 2.0 file decompression.
  Compatibility with later versions is not ensured as they may use
  unsupported compression algorithms.
  Versions after 2.7 may have an incompatible header format and thus be
  completely incompatible.
*/

/*! \enum UnZip::ErrorCode The result of a decompression operation.
  \value UnZip::Ok No error occurred.
  \value UnZip::ZlibInit Failed to init or load the zlib library.
  \value UnZip::ZlibError The zlib library returned some error.
  \value UnZip::OpenFailed Unable to create or open a device.
  \value UnZip::PartiallyCorrupted Corrupted zip archive - some files could be extracted.
  \value UnZip::Corrupted Corrupted or invalid zip archive.
  \value UnZip::WrongPassword Unable to decrypt a password protected file.
  \value UnZip::NoOpenArchive No archive has been opened yet.
  \value UnZip::FileNotFound Unable to find the requested file in the archive.
  \value UnZip::ReadFailed Reading of a file failed.
  \value UnZip::WriteFailed Writing of a file failed.
  \value UnZip::SeekFailed Seek failed.
  \value UnZip::CreateDirFailed Could not create a directory.
  \value UnZip::InvalidDevice A null device has been passed as parameter.
  \value UnZip::InvalidArchive This is not a valid (or supported) ZIP archive.
  \value UnZip::HeaderConsistencyError Local header record info does not match with the central directory record info. The archive may be corrupted.

  \value UnZip::Skip Internal use only.
  \value UnZip::SkipAll Internal use only.
*/
```

```

/*! \enum UnZip::ExtractionOptions Some options for the file extraction methods.
  \value UnZip::ExtractPaths Default. Does not ignore the path of the zipped files.
  \value UnZip::SkipPaths Default. Ignores the path of the zipped files and extracts them all to the same root directory.
*/

//! Local header size (excluding signature, excluding variable length fields)
#define UNZIP_LOCAL_HEADER_SIZE 26
//! Central Directory file entry size (excluding signature, excluding variable length fields)
#define UNZIP_CD_ENTRY_SIZE_NS 42
//! Data descriptor size (excluding signature)
#define UNZIP_DD_SIZE 12
//! End Of Central Directory size (including signature, excluding variable length fields)
#define UNZIP_EOCD_SIZE 22
//! Local header entry encryption header size
#define UNZIP_LOCAL_ENC_HEADER_SIZE 12

// Some offsets inside a CD record (excluding signature)
#define UNZIP_CD_OFF_VERSION 0
#define UNZIP_CD_OFF_GPFLAG 4
#define UNZIP_CD_OFF_CMETHOD 6
#define UNZIP_CD_OFF_MODT 8
#define UNZIP_CD_OFF_MODD 10
#define UNZIP_CD_OFF_CRC32 12
#define UNZIP_CD_OFF_CSIZE 16
#define UNZIP_CD_OFF_USIZE 20
#define UNZIP_CD_OFF_NAMELEN 24
#define UNZIP_CD_OFF_XLEN 26
#define UNZIP_CD_OFF_COMMLEN 28
#define UNZIP_CD_OFF_LHOFFSET 38

// Some offsets inside a local header record (excluding signature)
#define UNZIP_LH_OFF_VERSION 0
#define UNZIP_LH_OFF_GPFLAG 2
#define UNZIP_LH_OFF_CMETHOD 4
#define UNZIP_LH_OFF_MODT 6
#define UNZIP_LH_OFF_MODD 8
#define UNZIP_LH_OFF_CRC32 10
#define UNZIP_LH_OFF_CSIZE 14
#define UNZIP_LH_OFF_USIZE 18
#define UNZIP_LH_OFF_NAMELEN 22
#define UNZIP_LH_OFF_XLEN 24

// Some offsets inside a data descriptor record (excluding signature)
#define UNZIP_DD_OFF_CRC32 0
#define UNZIP_DD_OFF_CSIZE 4
#define UNZIP_DD_OFF_USIZE 8

// Some offsets inside a EOCD record
#define UNZIP_EOCD_OFF_ENTRIES 6
#define UNZIP_EOCD_OFF_CDOFF 12
#define UNZIP_EOCD_OFF_COMMLEN 16

/*!
  Max version handled by this API.
  0x1B = 2.7 --> full compatibility only up to version 2.0 (0x14)
  versions from 2.1 to 2.7 may use unsupported compression methods
  versions after 2.7 may have an incompatible header format
*/
#define UNZIP_VERSION 0x1B
//! Full compatibility granted until this version
#define UNZIP_VERSION_STRICT 0x14

//! CRC32 routine
#define CRC32(c, b) crcTable[((int)c^b) & 0xff] ^ (c >> 8)

//! Checks if some file has been already extracted.
#define UNZIP_CHECK_FOR_VALID_DATA \
  {\
    if (headers != 0)\
    {\
      qDebug() << "Corrupted zip archive. Some files might be extracted.";\
      ec = headers->size() != 0 ? UnZip::PartiallyCorrupted : UnZip::Corrupted;\
      break;\
    }\
    else\
    {\
      delete device;\
      device = 0;\
      qDebug() << "Corrupted or invalid zip archive";\
      ec = UnZip::Corrupted;\
      break;\
    }\
  }


```

```
/************************************************************************
 Public interface
 ************************************************************************/

/*!
  Creates a new Zip file decompressor.
*/
UnZip::UnZip()
{
  d = new UnzipPrivate;
}

/*!
  Closes any open archive and releases used resources.
*/
UnZip::~UnZip()
{
  closeArchive();
  delete d;
}

/*!
  Returns true if there is an open archive.
*/
bool UnZip::isOpen() const
{
  return d->device != 0;
}

/*!
  Opens a zip archive and reads the files list. Closes any previously opened archive.
*/
UnZip::ErrorCode UnZip::openArchive(const QString& filename)
{
  QFile* file = new QFile(filename);

  if (!file->exists()) {
    delete file;
    return UnZip::FileNotFound;
  }

  if (!file->open(QIODevice::ReadOnly)) {
    delete file;
    return UnZip::OpenFailed;
  }

  return openArchive(file);
}

/*!
  Opens a zip archive and reads the entries list.
  Closes any previously opened archive.
  \warning The class takes ownership of the device so don't delete it!
*/
UnZip::ErrorCode UnZip::openArchive(QIODevice* device)
{
  if (device == 0)
  {
    qDebug() << "Invalid device.";
    return UnZip::InvalidDevice;
  }

  return d->openArchive(device);
}

/*!
  Closes the archive and releases all the used resources (like cached passwords).
*/
void UnZip::closeArchive()
{
  d->closeArchive();
}

QString UnZip::archiveComment() const
{
  if (d->device == 0)
    return QString();
  return d->comment;
}

/*!
  Returns a locale translated error string for a given error code.
*/
QString UnZip::formatError(UnZip::ErrorCode c) const
{
  switch (c)
  {
```

```
  case Ok: return QCoreApplication::translate("UnZip", "ZIP operation completed successfully."); break;
  case ZlibInit: return QCoreApplication::translate("UnZip", "Failed to initialize or load zlib
 library."); break;
  case ZlibError: return QCoreApplication::translate("UnZip", "zlib library error."); break;
  case OpenFailed: return QCoreApplication::translate("UnZip", "Unable to create or open file.");
 break;
  case PartiallyCorrupted: return QCoreApplication::translate("UnZip", "Partially
 corrupted archive. Some files might be extracted."); break;
  case Corrupted: return QCoreApplication::translate("UnZip", "Corrupted archive."); break;
  case WrongPassword: return QCoreApplication::translate("UnZip", "Wrong password."); break;
  case NoOpenArchive: return QCoreApplication::translate("UnZip", "No archive has been created
 yet."); break;
  case FileNotFound: return QCoreApplication::translate("UnZip", "File or directory does not
 exist."); break;
  case ReadFailed: return QCoreApplication::translate("UnZip", "File read error."); break;
  case WriteFailed: return QCoreApplication::translate("UnZip", "File write error."); break;
  case SeekFailed: return QCoreApplication::translate("UnZip", "File seek error."); break;
  case CreateDirFailed: return QCoreApplication::translate("UnZip", "Unable to create a
 directory."); break;
  case InvalidDevice: return QCoreApplication::translate("UnZip", "Invalid device."); break;
  case InvalidArchive: return QCoreApplication::translate("UnZip", "Invalid or incompatible
 zip archive."); break;
  case HeaderConsistencyError: return QCoreApplication::translate("UnZip", "
Inconsistent headers. Archive might be corrupted."); break;
  default: ;
  }

  return QCoreApplication::translate("UnZip", "Unknown error.");
}

/*!
  Returns true if the archive contains a file with the given path and name.
*/
bool UnZip::contains(const QString& file) const
{
  if (d->headers == 0)
    return false;

  return d->headers->contains(file);
}

/*!
  Returns complete paths of files and directories in this archive.
*/
QStringList UnZip::fileList() const
{
  return d->headers == 0 ? QStringList() : d->headers->keys();
}

/*!
  Returns information for each (correctly parsed) entry of this archive.
*/
QList<UnZip::ZipEntry> UnZip::entryList() const
{
  QList<UnZip::ZipEntry> list;

  if (d->headers != 0)
  {
    for (QMap<QString,ZipEntryP*>::ConstIterator it = d->headers->constBegin(); it !=
d->headers->constEnd(); ++it)
    {
      const ZipEntryP* entry = it.value();
      Q_ASSERT(entry != 0);

      ZipEntry z;

      z.filename = it.key();
      if (!entry->comment.isEmpty())
        z.comment = entry->comment;
      z.compressedSize = entry->szComp;
      z.uncompressedSize = entry->szUncomp;
      z.crc32 = entry->crc;
      z.lastModified = d->convertDateTime(entry->
modDate, entry->modTime);

      z.compression = entry->compMethod == 0 ?
NoCompression : entry->compMethod == 8 ? Deflated :
UnknownCompression;
      z.type = z.filename.endsWith("/") ? Directory : File;

      z.encrypted = entry->isEncrypted();

      list.append(z);
    }
  }

  return list;
```

```cpp
}

/*!
  Extracts the whole archive to a directory.
*/
UnZip::ErrorCode UnZip::extractAll(const QString& dirname, ExtractionOptions options)
{
  return extractAll(QDir(dirname), options);
}

/*!
  Extracts the whole archive to a directory.
*/
UnZip::ErrorCode UnZip::extractAll(const QDir& dir, ExtractionOptions options)
{
  // this should only happen if we didn’t call openArchive() yet
  if (d->device == 0)
    return NoOpenArchive;

  if (d->headers == 0)
    return Ok;

  bool end = false;
  for (QMap<QString,ZipEntryP*>::Iterator itr = d->headers->begin(); itr != d->headers->end(); ++itr)
  {
    ZipEntryP* entry = itr.value();
    Q_ASSERT(entry != 0);

    if ((entry->isEncrypted()) && d->skipAllEncrypted)
      continue;

    switch (d->extractFile(itr.key(), *entry, dir, options))
    {
    case Corrupted:
      qDebug() << "Removing corrupted entry" << itr.key();
      d->headers->erase(itr++);
      if (itr == d->headers->end())
        end = true;
      break;
    case CreateDirFailed:
      break;
    case Skip:
      break;
    case SkipAll:
      d->skipAllEncrypted = true;
      break;
    default:
      ;
    }

    if (end)
      break;
  }

  return Ok;
}

/*!
  Extracts a single file to a directory.
*/
UnZip::ErrorCode UnZip::extractFile(const QString& filename, const QString& dirname, ExtractionOptions options)
{
  return extractFile(filename, QDir(dirname), options);
}

/*!
  Extracts a single file to a directory.
*/
UnZip::ErrorCode UnZip::extractFile(const QString& filename, const QDir& dir, ExtractionOptions options)
{
  QMap<QString,ZipEntryP*>::Iterator itr = d->headers->find(filename);
  if (itr != d->headers->end())
  {
    ZipEntryP* entry = itr.value();
    Q_ASSERT(entry != 0);
    return d->extractFile(itr.key(), *entry, dir, options);
  }

  return FileNotFound;
}

/*!
```

```
  Extracts a single file to a directory.
*/
UnZip::ErrorCode UnZip::extractFile(const QString& filename, QIODevice* dev, ExtractionOptions options)
{
  if (dev == 0)
    return InvalidDevice;

  QMap<QString,ZipEntryP*>::Iterator itr = d->headers->find(filename);
  if (itr != d->headers->end()) {
    ZipEntryP* entry = itr.value();
    Q_ASSERT(entry != 0);
    return d->extractFile(itr.key(), *entry, dev, options);
  }

  return FileNotFound;
}

/*!
  Extracts a list of files.
  Stops extraction at the first error (but continues if a file does not exist in the archive).
 */
UnZip::ErrorCode UnZip::extractFiles(const QStringList& filenames, const QString& dirname, ExtractionOptions options)
{
  QDir dir(dirname);
  ErrorCode ec;

  for (QStringList::ConstIterator itr = filenames.constBegin(); itr != filenames.constEnd(); ++itr)
  {
    ec = extractFile(*itr, dir, options);
    if (ec == FileNotFound)
      continue;
    if (ec != Ok)
      return ec;
  }

  return Ok;
}

/*!
  Extracts a list of files.
  Stops extraction at the first error (but continues if a file does not exist in the archive).
 */
UnZip::ErrorCode UnZip::extractFiles(const QStringList& filenames, const QDir& dir, ExtractionOptions options)
{
  ErrorCode ec;

  for (QStringList::ConstIterator itr = filenames.constBegin(); itr != filenames.constEnd(); ++itr)
  {
    ec = extractFile(*itr, dir, options);
    if (ec == FileNotFound)
      continue;
    if (ec != Ok)
      return ec;
  }

  return Ok;
}

/*!
  Remove/replace this method to add your own password retrieval routine.
*/
void UnZip::setPassword(const QString& pwd)
{
  d->password = pwd;
}

/*!
  ZipEntry constructor - initialize data. Type is set to File.
*/
UnZip::ZipEntry::ZipEntry()
{
  compressedSize = uncompressedSize = crc32 = 0;
  compression = NoCompression;
  type = File;
  encrypted = false;
}


/************************************************************************
 Private interface
*************************************************************************/

//! \internal
UnzipPrivate::UnzipPrivate()
```

```
{
  skipAllEncrypted = false;
  headers = 0;
  device = 0;

  uBuffer = (unsigned char*) buffer1;
  crcTable = (quint32*) get_crc_table();

  cdOffset = eocdOffset = 0;
  cdEntryCount = 0;
  unsupportedEntryCount = 0;
}

//! \internal Parses a Zip archive.
UnZip::ErrorCode UnzipPrivate::openArchive(QIODevice* dev)
{
  Q_ASSERT(dev != 0);

  if (device != 0)
    closeArchive();

  device = dev;

  if (!(device->isOpen() || device->open(QIODevice::ReadOnly)))
  {
    delete device;
    device = 0;

    qDebug() << "Unable to open device for reading";
    return UnZip::OpenFailed;
  }

  UnZip::ErrorCode ec;

  ec = seekToCentralDirectory();
  if (ec != UnZip::Ok)
  {
    closeArchive();
    return ec;
  }

  //! \todo Ignore CD entry count? CD may be corrupted.
  if (cdEntryCount == 0)
  {
    return UnZip::Ok;
  }

  bool continueParsing = true;

  while (continueParsing)
  {
    if (device->read(buffer1, 4) != 4)
      UNZIP_CHECK_FOR_VALID_DATA

    if (! (buffer1[0] == ’P’ && buffer1[1] == ’K’ && buffer1[2] == 0x01  && buffer1[3] == 0x02) )
      break;

    if ( (ec = parseCentralDirectoryRecord()) != UnZip::Ok )
      break;
  }

  if (ec != UnZip::Ok)
    closeArchive();

  return ec;
}

/*
  \internal Parses a local header record and makes some consistency check
  with the information stored in the Central Directory record for this entry
  that has been previously parsed.
  \todo Optional consistency check (as a ExtractionOptions flag)

  local file header signature     4 bytes  (0x04034b50)
  version needed to extract       2 bytes
  general purpose bit flag        2 bytes
  compression method              2 bytes
  last mod file time              2 bytes
  last mod file date              2 bytes
  crc-32                          4 bytes
  compressed size                 4 bytes
  uncompressed size               4 bytes
  file name length                2 bytes
  extra field length              2 bytes

  file name (variable size)
  extra field (variable size)
```

```
*/
UnZip::ErrorCode UnzipPrivate::parseLocalHeaderRecord(const QString& path, ZipEntryP& entry)
{
  if (!device->seek(entry.lhOffset))
    return UnZip::SeekFailed;

  // Test signature
  if (device->read(buffer1, 4) != 4)
    return UnZip::ReadFailed;

  if ((buffer1[0] != 'P') || (buffer1[1] != 'K') || (buffer1[2] != 0x03) || (buffer1[3] != 0x04))
    return UnZip::InvalidArchive;

  if (device->read(buffer1, UNZIP_LOCAL_HEADER_SIZE) != UNZIP_LOCAL_HEADER_SIZE)
    return UnZip::ReadFailed;

  /*
    Check 3rd general purpose bit flag.

    "bit 3: If this bit is set, the fields crc-32, compressed size
    and uncompressed size are set to zero in the local
    header.  The correct values are put in the data descriptor
    immediately following the compressed data."
  */
  bool hasDataDescriptor = entry.hasDataDescriptor();

  bool checkFailed = false;

  if (!checkFailed)
    checkFailed = entry.compMethod != getUShort(uBuffer, UNZIP_LH_OFF_CMETHOD);
  if (!checkFailed)
    checkFailed = entry.gpFlag[0] != uBuffer[UNZIP_LH_OFF_GPFLAG];
  if (!checkFailed)
    checkFailed = entry.gpFlag[1] != uBuffer[UNZIP_LH_OFF_GPFLAG + 1];
  if (!checkFailed)
    checkFailed = entry.modTime[0] != uBuffer[UNZIP_LH_OFF_MODT];
  if (!checkFailed)
    checkFailed = entry.modTime[1] != uBuffer[UNZIP_LH_OFF_MODT + 1];
  if (!checkFailed)
    checkFailed = entry.modDate[0] != uBuffer[UNZIP_LH_OFF_MODD];
  if (!checkFailed)
    checkFailed = entry.modDate[1] != uBuffer[UNZIP_LH_OFF_MODD + 1];
  if (!hasDataDescriptor)
  {
    if (!checkFailed)
      checkFailed = entry.crc != getULong(uBuffer, UNZIP_LH_OFF_CRC32);
    if (!checkFailed)
      checkFailed = entry.szComp != getULong(uBuffer, UNZIP_LH_OFF_CSIZE);
    if (!checkFailed)
      checkFailed = entry.szUncomp != getULong(uBuffer, UNZIP_LH_OFF_USIZE);
  }

  if (checkFailed)
    return UnZip::HeaderConsistencyError;

  // Check filename
  quint16 szName = getUShort(uBuffer, UNZIP_LH_OFF_NAMELEN);
  if (szName == 0)
    return UnZip::HeaderConsistencyError;

  if (device->read(buffer2, szName) != szName)
    return UnZip::ReadFailed;

    //QString filename = QString::fromAscii(buffer2, szName); // Qt4
    QString filename = QString::fromLatin1(buffer2, szName);  // Qt5
  if (filename != path)
  {
    qDebug() << "Filename in local header mismatches.";
    return UnZip::HeaderConsistencyError;
  }

  // Skip extra field
  quint16 szExtra = getUShort(uBuffer, UNZIP_LH_OFF_XLEN);
  if (szExtra != 0)
  {
    if (!device->seek(device->pos() + szExtra))
      return UnZip::SeekFailed;
  }

  entry.dataOffset = device->pos();

  if (hasDataDescriptor)
  {
    /*
      The data descriptor has this OPTIONAL signature: PK\7\8
      We try to skip the compressed data relying on the size set in the
      Central Directory record.
    */
    if (!device->seek(device->pos() + entry.szComp))
      return UnZip::SeekFailed;

    // Read 4 bytes and check if there is a data descriptor signature
    if (device->read(buffer2, 4) != 4)
      return UnZip::ReadFailed;

    bool hasSignature = buffer2[0] == 'P' && buffer2[1] == 'K' && buffer2[2] == 0x07 && buffer2[3] == 0x08;
    if (hasSignature)
    {
      if (device->read(buffer2, UNZIP_DD_SIZE) != UNZIP_DD_SIZE)
        return UnZip::ReadFailed;
    }
    else
    {
      if (device->read(buffer2 + 4, UNZIP_DD_SIZE - 4) !=
UNZIP_DD_SIZE - 4)
        return UnZip::ReadFailed;
    }

    // DD: crc, compressed size, uncompressed size
    if (
      entry.crc != getULong((unsigned char*)buffer2, UNZIP_DD_OFF_CRC32) ||
      entry.szComp != getULong((unsigned char*)buffer2, UNZIP_DD_OFF_CSIZE) ||
      entry.szUncomp != getULong((unsigned char*)buffer2,
UNZIP_DD_OFF_USIZE)
      )
      return UnZip::HeaderConsistencyError;
  }

  return UnZip::Ok;
}

/*! \internal Attempts to find the start of the central directory record.

  We seek the file back until we reach the "End Of Central Directory"
  signature PK\5\6.

  end of central dir signature    4 bytes  (0x06054b50)
  number of this disk             2 bytes
  number of the disk with the
  start of the central directory  2 bytes
  total number of entries in the
  central directory on this disk  2 bytes
  total number of entries in
  the central directory           2 bytes
  size of the central directory   4 bytes
  offset of start of central
  directory with respect to
  the starting disk number        4 bytes
  .ZIP file comment length        2 bytes
  --- SIZE UNTIL HERE: UNZIP_EOCD_SIZE ---
  .ZIP file comment       (variable size)
*/
UnZip::ErrorCode UnzipPrivate::seekToCentralDirectory()
{
  qint64 length = device->size();
  qint64 offset = length - UNZIP_EOCD_SIZE;

  if (length < UNZIP_EOCD_SIZE)
    return UnZip::InvalidArchive;

  if (!device->seek( offset ))
    return UnZip::SeekFailed;

  if (device->read(buffer1, UNZIP_EOCD_SIZE) != UNZIP_EOCD_SIZE)
    return UnZip::ReadFailed;

  bool eocdFound = (buffer1[0] == 'P' && buffer1[1] == 'K' && buffer1[2] == 0x05 && buffer1[3] == 0x06);

  if (eocdFound)
  {
    // Zip file has no comment (the only variable length field in the EOCD record)
    eocdOffset = offset;
  }
  else
  {
    qint64 read;
    char* p = 0;

    offset -= UNZIP_EOCD_SIZE;

  if (offset <= 0)
   return UnZip::InvalidArchive;

  if (!device->seek( offset ))
   return UnZip::SeekFailed;

  while ((read = device->read(buffer1, UNZIP_EOCD_SIZE)) >= 0)
  {
   if ( (p = strstr(buffer1, "PK\5\6")) != 0)
   {
    // Seek to the start of the EOCD record so we can read it fully
    // Yes... we could simply read the missing bytes and append them to the buffer
    // but this is far easier so heck it!
    device->seek( offset + (p - buffer1) );
    eocdFound = true;
    eocdOffset = offset + (p - buffer1);

    // Read EOCD record
    if (device->read(buffer1, UNZIP_EOCD_SIZE) != UNZIP_EOCD_SIZE)
     return UnZip::ReadFailed;

    break;
   }

   offset -= UNZIP_EOCD_SIZE;
   if (offset <= 0)
    return UnZip::InvalidArchive;

   if (!device->seek( offset ))
    return UnZip::SeekFailed;
  }
 }

 if (!eocdFound)
  return UnZip::InvalidArchive;

 // Parse EOCD to locate CD offset
 offset = getULong((const unsigned char*)buffer1, UNZIP_EOCD_OFF_CDOFF + 4);

 cdOffset = offset;

 cdEntryCount = getUShort((const unsigned char*)buffer1, UNZIP_EOCD_OFF_ENTRIES + 4)
;

 quint16 commentLength = getUShort((const unsigned char*)buffer1,
UNZIP_EOCD_OFF_COMMLEN + 4);
 if (commentLength != 0)
 {
  QByteArray c = device->read(commentLength);
  if (c.count() != commentLength)
   return UnZip::ReadFailed;

  comment = c;
 }

 // Seek to the start of the CD record
 if (!device->seek( cdOffset ))
  return UnZip::SeekFailed;

 return UnZip::Ok;
}

/*!
 \internal Parses a central directory record.

 Central Directory record structure:

 [file header 1]
 .
 .
 .
 [file header n]
 [digital signature] // PKZip 6.2 or later only

 File header:

 central file header signature   4 bytes  (0x02014b50)
 version made by                 2 bytes
 version needed to extract       2 bytes
 general purpose bit flag        2 bytes
 compression method              2 bytes
 last mod file time              2 bytes
 last mod file date              2 bytes
 crc-32                          4 bytes
 compressed size                 4 bytes
 uncompressed size               4 bytes
 file name length                2 bytes
```

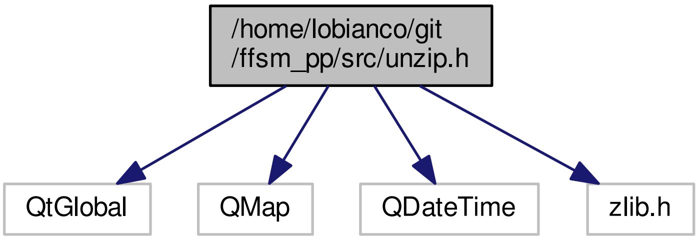
```
  extra field length              2 bytes
  file comment length             2 bytes
  disk number start               2 bytes
  internal file attributes        2 bytes
  external file attributes        4 bytes
  relative offset of local header 4 bytes

  file name (variable size)
  extra field (variable size)
  file comment (variable size)
*/
UnZip::ErrorCode UnzipPrivate::parseCentralDirectoryRecord
()
{
  // Read CD record
  if (device->read(buffer1, UNZIP_CD_ENTRY_SIZE_NS) !=
UNZIP_CD_ENTRY_SIZE_NS)
    return UnZip::ReadFailed;

  bool skipEntry = false;

  // Get compression type so we can skip non compatible algorithms
  quint16 compMethod = getUShort(uBuffer, UNZIP_CD_OFF_CMETHOD);

  // Get variable size fields length so we can skip the whole record
  // if necessary
  quint16 szName = getUShort(uBuffer, UNZIP_CD_OFF_NAMELEN);
  quint16 szExtra = getUShort(uBuffer, UNZIP_CD_OFF_XLEN);
  quint16 szComment = getUShort(uBuffer, UNZIP_CD_OFF_COMMLEN);

  quint32 skipLength = szName + szExtra + szComment;

  UnZip::ErrorCode ec = UnZip::Ok;

  if ((compMethod != 0) && (compMethod != 8))
  {
    qDebug() << "Unsupported compression method. Skipping file.";
    skipEntry = true;
  }

  // Header parsing may be a problem if version is bigger than UNZIP_VERSION
  if (!skipEntry && buffer1[UNZIP_CD_OFF_VERSION] >
UNZIP_VERSION)
  {
    qDebug() << "Unsupported PKZip version. Skipping file.";
    skipEntry = true;
  }

  if (!skipEntry && szName == 0)
  {
    qDebug() << "Skipping file with no name.";
    skipEntry = true;
  }

  if (!skipEntry && device->read(buffer2, szName) != szName)
  {
    ec = UnZip::ReadFailed;
    skipEntry = true;
  }

  if (skipEntry)
  {
    if (ec == UnZip::Ok)
    {
      if (!device->seek( device->pos() + skipLength ))
        ec = UnZip::SeekFailed;

      unsupportedEntryCount++;
    }

    return ec;
  }

    //QString filename = QString::fromAscii(buffer2, szName); // Qt4
    QString filename = QString::fromLatin1(buffer2, szName);  // Qt5

  ZipEntryP* h = new ZipEntryP;
  h->compMethod = compMethod;

  h->gpFlag[0] = buffer1[UNZIP_CD_OFF_GPFLAG];
  h->gpFlag[1] = buffer1[UNZIP_CD_OFF_GPFLAG + 1];

  h->modTime[0] = buffer1[UNZIP_CD_OFF_MODT];
  h->modTime[1] = buffer1[UNZIP_CD_OFF_MODT + 1];

  h->modDate[0] = buffer1[UNZIP_CD_OFF_MODD];
  h->modDate[1] = buffer1[UNZIP_CD_OFF_MODD + 1];

  h->crc = getULong(uBuffer, UNZIP_CD_OFF_CRC32);
  h->szComp = getULong(uBuffer, UNZIP_CD_OFF_CSIZE);
  h->szUncomp = getULong(uBuffer, UNZIP_CD_OFF_USIZE);

  // Skip extra field (if any)
  if (szExtra != 0)
  {
    if (!device->seek( device->pos() + szExtra ))
    {
      delete h;
      return UnZip::SeekFailed;
    }
  }

  // Read comment field (if any)
  if (szComment != 0)
  {
    if (device->read(buffer2, szComment) != szComment)
    {
      delete h;
      return UnZip::ReadFailed;
    }

        //h->comment = QString::fromAscii(buffer2, szComment); // Qt4
        h->comment = QString::fromLatin1(buffer2, szComment);  // Qt5
  }

  h->lhOffset = getULong(uBuffer, UNZIP_CD_OFF_LHOFFSET);

  if (headers == 0)
    headers = new QMap<QString, ZipEntryP*>();
  headers->insert(filename, h);

  return UnZip::Ok;
}

//! \internal Closes the archive and resets the internal status.
void UnzipPrivate::closeArchive()
{
  if (device == 0)
    return;

  skipAllEncrypted = false;

  if (headers != 0)
  {
    qDeleteAll(*headers);
    delete headers;
    headers = 0;
  }

  delete device; device = 0;

  cdOffset = eocdOffset = 0;
  cdEntryCount = 0;
  unsupportedEntryCount = 0;

  comment.clear();
}

//! \internal
UnZip::ErrorCode UnzipPrivate::extractFile(const QString& path, ZipEntryP& entry, const QDir& dir, UnZip::ExtractionOptions options)
{
  QString name(path);
  QString dirname;
  QString directory;

  int pos = name.lastIndexOf('/');

  // This entry is for a directory
  if (pos == name.length() - 1)
  {
    if (options.testFlag(UnZip::SkipPaths))
      return UnZip::Ok;

    directory = QString("%1/%2").arg(dir.absolutePath()).arg(QDir::cleanPath(name));
    if (!createDirectory(directory))
    {
      qDebug() << QString("Unable to create directory: %1").arg(directory);
      return UnZip::CreateDirFailed;
    }

    return UnZip::Ok;
  }

  // Extract path from entry
  if (pos > 0)
  {
    // get directory part
    dirname = name.left(pos);
    if (options.testFlag(UnZip::SkipPaths))
    {
      directory = dir.absolutePath();
    }
    else
    {
      directory = QString("%1/%2").arg(dir.absolutePath()).arg(QDir::cleanPath(dirname));
      if (!createDirectory(directory))
      {
        qDebug() << QString("Unable to create directory: %1").arg(directory);
        return UnZip::CreateDirFailed;
      }
    }
    name = name.right(name.length() - pos - 1);
  } else directory = dir.absolutePath();

  name = QString("%1/%2").arg(directory).arg(name);

  QFile outFile(name);

  if (!outFile.open(QIODevice::WriteOnly))
  {
    qDebug() << QString("Unable to open %1 for writing").arg(name);
    return UnZip::OpenFailed;
  }

  //! \todo Set creation/last_modified date/time

  UnZip::ErrorCode ec = extractFile(path, entry, &outFile, options);

  outFile.close();

  if (ec != UnZip::Ok)
  {
    if (!outFile.remove())
      qDebug() << QString("Unable to remove corrupted file: %1").arg(name);
  }

  return ec;
}

//! \internal
UnZip::ErrorCode UnzipPrivate::extractFile(const QString& path, ZipEntryP& entry, QIODevice* dev, UnZip::ExtractionOptions options)
{
  Q_UNUSED(options);
  Q_ASSERT(dev != 0);

  if (!entry.lhEntryChecked)
  {
    UnZip::ErrorCode ec = parseLocalHeaderRecord(path, entry);
    entry.lhEntryChecked = true;

    if (ec != UnZip::Ok)
      return ec;
  }

  if (!device->seek(entry.dataOffset))
    return UnZip::SeekFailed;

  // Encryption keys
  quint32 keys[3];

  if (entry.isEncrypted())
  {
    UnZip::ErrorCode e = testPassword(keys, path, entry);
    if (e != UnZip::Ok)
    {
      qDebug() << QString("Unable to decrypt %1").arg(path);
      return e;
    }//! Encryption header size
    entry.szComp -= UNZIP_LOCAL_ENC_HEADER_SIZE; // remove encryption header size
  }

  if (entry.szComp == 0)
  {
    if (entry.crc != 0)
      return UnZip::Corrupted;

    return UnZip::Ok;
  }

  uInt rep = entry.szComp / UNZIP_READ_BUFFER;
  uInt rem = entry.szComp % UNZIP_READ_BUFFER;
  uInt cur = 0;

  // extract data
  qint64 read;
  quint64 tot = 0;

  quint32 myCRC = crc32(0L, Z_NULL, 0);

  if (entry.compMethod == 0)
  {
    while ( (read = device->read(buffer1, cur < rep ? UNZIP_READ_BUFFER : rem)) > 0 )
    {
      if (entry.isEncrypted())
        decryptBytes(keys, buffer1, read);

      myCRC = crc32(myCRC, uBuffer, read);

      if (dev->write(buffer1, read) != read)
        return UnZip::WriteFailed;

      cur++;
      tot += read;

      if (tot == entry.szComp)
        break;
    }

    if (read < 0)
      return UnZip::ReadFailed;
  }
  else if (entry.compMethod == 8)
  {
    /* Allocate inflate state */
    z_stream zstr;
    zstr.zalloc = Z_NULL;
    zstr.zfree = Z_NULL;
    zstr.opaque = Z_NULL;
    zstr.next_in = Z_NULL;
    zstr.avail_in = 0;

    int zret;

    // Use inflateInit2 with negative windowBits to get raw decompression
    if ( (zret = inflateInit2_(&zstr, -MAX_WBITS, ZLIB_VERSION, sizeof(z_stream))) != Z_OK )
      return UnZip::ZlibError;

    int szDecomp;

    // Decompress until deflate stream ends or end of file
    do
    {
      read = device->read(buffer1, cur < rep ? UNZIP_READ_BUFFER : rem);
      if (read == 0)
        break;
      if (read < 0)
      {
        (void)inflateEnd(&zstr);
        return UnZip::ReadFailed;
      }

      if (entry.isEncrypted())
        decryptBytes(keys, buffer1, read);

      cur++;
      tot += read;

      zstr.avail_in = (uInt) read;
      zstr.next_in = (Bytef*) buffer1;


      // Run inflate() on input until output buffer not full
      do {
        zstr.avail_out = UNZIP_READ_BUFFER;
        zstr.next_out = (Bytef*) buffer2;;

        zret = inflate(&zstr, Z_NO_FLUSH);

        switch (zret) {
          case Z_NEED_DICT:
          case Z_DATA_ERROR:
          case Z_MEM_ERROR:
            inflateEnd(&zstr);
            return UnZip::WriteFailed;
          default:
            ;
        }

        szDecomp = UNZIP_READ_BUFFER - zstr.avail_out;
        if (dev->write(buffer2, szDecomp) != szDecomp)
        {
          inflateEnd(&zstr);
          return UnZip::ZlibError;
        }

        myCRC = crc32(myCRC, (const Bytef*) buffer2, szDecomp);

      } while (zstr.avail_out == 0);

    }
    while (zret != Z_STREAM_END);

    inflateEnd(&zstr);
  }

  if (myCRC != entry.crc)
    return UnZip::Corrupted;

  return UnZip::Ok;
}

//! \internal Creates a new directory and all the needed parent directories.
bool UnzipPrivate::createDirectory(const QString& path)
{
  QDir d(path);
  if (!d.exists())
  {
    int sep = path.lastIndexOf("/");
    if (sep <= 0) return true;

    if (!createDirectory(path.left(sep)))
      return false;

    if (!d.mkdir(path))
    {
      qDebug() << QString("Unable to create directory: %1").arg(path);
      return false;
    }
  }

  return true;
}

/*!
  \internal Reads an quint32 (4 bytes) from a byte array starting at given offset.
*/
quint32 UnzipPrivate::getULong(const unsigned char* data, quint32 offset) const
{
  quint32 res = (quint32) data[offset];
  res |= (((quint32)data[offset+1]) << 8);
  res |= (((quint32)data[offset+2]) << 16);
  res |= (((quint32)data[offset+3]) << 24);

  return res;
}

/*!
  \internal Reads an quint64 (8 bytes) from a byte array starting at given offset.
*/
quint64 UnzipPrivate::getULLong(const unsigned char* data, quint32 offset) const
{
  quint64 res = (quint64) data[offset];
  res |= (((quint64)data[offset+1]) << 8);
  res |= (((quint64)data[offset+2]) << 16);
  res |= (((quint64)data[offset+3]) << 24);
  res |= (((quint64)data[offset+1]) << 32);
  res |= (((quint64)data[offset+2]) << 40);
  res |= (((quint64)data[offset+3]) << 48);
  res |= (((quint64)data[offset+3]) << 56);

  return res;
}

/*!
  \internal Reads an quint16 (2 bytes) from a byte array starting at given offset.
*/
quint16 UnzipPrivate::getUShort(const unsigned char* data, quint32 offset) const
{
  return (quint16) data[offset] | (((quint16)data[offset+1]) << 8);
}

/*!
```

```
  \internal Return the next byte in the pseudo-random sequence
 */
int UnzipPrivate::decryptByte(quint32 key2) const
{
  quint16 temp = ((quint16)(key2) & 0xffff) | 2;
  return (int)(((temp * (temp ^ 1)) >> 8) & 0xff);
}

/*!
  \internal Update the encryption keys with the next byte of plain text
 */
void UnzipPrivate::updateKeys(quint32* keys, int c) const
{
  keys[0] = CRC32(keys[0], c);
  keys[1] += keys[0] & 0xff;
  keys[1] = keys[1] * 134775813L + 1;
  keys[2] = CRC32(keys[2], ((int)keys[1]) >> 24);
}

/*!
  \internal Initialize the encryption keys and the random header according to
  the given password.
 */
void UnzipPrivate::initKeys(const QString& pwd, quint32* keys) const
{
  keys[0] = 305419896L;
  keys[1] = 591751049L;
  keys[2] = 878082192L;

    //QByteArray pwdBytes = pwd.toAscii(); // Qt4
    QByteArray pwdBytes = pwd.toLatin1();  // Qt5
  int sz = pwdBytes.size();
  const char* ascii = pwdBytes.data();

  for (int i=0; i<sz; ++i)
    updateKeys(keys, (int)ascii[i]);
}

/*!
  \internal Attempts to test a password without actually extracting a file.
  The \p file parameter can be used in the user interface or for debugging purposes
  as it is the name of the encrypted file for wich the password is being tested.
*/
UnZip::ErrorCode UnzipPrivate::testPassword(quint32* keys, const QString& file, const ZipEntryP& header)
{
  Q_UNUSED(file);

  // read encryption keys
  if (device->read(buffer1, 12) != 12)
    return UnZip::Corrupted;

  // Replace this code if you want to i.e. call some dialog and ask the user for a password
  initKeys(password, keys);
  if (testKeys(header, keys))
    return UnZip::Ok;

  return UnZip::Skip;
}

/*!
  \internal Tests a set of keys on the encryption header.
*/
bool UnzipPrivate::testKeys(const ZipEntryP& header, quint32* keys)
{
  char lastByte;

  // decrypt encryption header
  for (int i=0; i<11; ++i)
    updateKeys(keys, lastByte = buffer1[i] ^ decryptByte(keys[2]));
  updateKeys(keys, lastByte = buffer1[11] ^ decryptByte(keys[2]));

  // if there is an extended header (bit in the gp flag) buffer[11] is a byte from the file time
  // with no extended header we have to check the crc high-order byte
  char c = ((header.gpFlag[0] & 0x08) == 8) ? header.modTime[1] : header.crc >> 24;

  return (lastByte == c);
}

/*!
  \internal Decrypts an array of bytes long \p read.
*/
void UnzipPrivate::decryptBytes(quint32* keys, char* buffer, qint64 read)
{
  for (int i=0; i<(int)read; ++i)
    updateKeys(keys, buffer[i] ^= decryptByte(keys[2]));
}

/*!
  \internal Converts date and time values from ZIP format to a QDateTime object.
*/
QDateTime UnzipPrivate::convertDateTime(const unsigned char date[2], const
unsigned char time[2]) const
{
  QDateTime dt;

  // Usual PKZip low-byte to high-byte order

  // Date: 7 bits = years from 1980, 4 bits = month, 5 bits = day
  quint16 year = (date[1] >> 1) & 127;
  quint16 month = ((date[1] << 3) & 14) | ((date[0] >> 5) & 7);
  quint16 day = date[0] & 31;

  // Time: 5 bits hour, 6 bits minutes, 5 bits seconds with a 2sec precision
  quint16 hour = (time[1] >> 3) & 31;
  quint16 minutes = ((time[1] << 3) & 56) | ((time[0] >> 5) & 7);
  quint16 seconds = (time[0] & 31) * 2;

  dt.setDate(QDate(1980 + year, month, day));
  dt.setTime(QTime(hour, minutes, seconds));
  return dt;
}
```

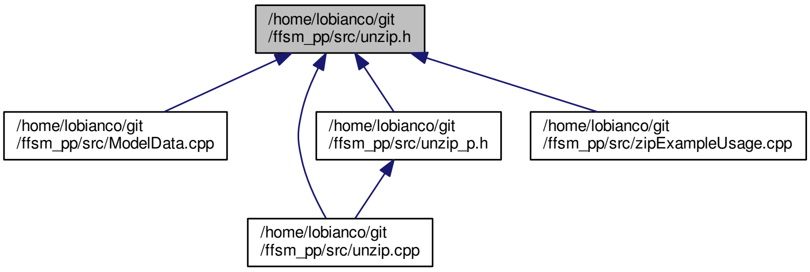
5.145 /home/lobianco/git/ffsm_pp/src/unzip.h File Reference

```
#include <QtGlobal>
#include <QMap>
#include <QDateTime>
#include <zlib.h>
```

Include dependency graph for unzip.h:



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



Classes

- class UnZip

  PKZip 2.0 file decompression. Compatibility with later versions is not ensured as they may use unsupported compression algorithms. Versions after 2.7 may have an incompatible header format and thus be completely incompatible.
- struct UnZip::ZipEntry

5.146 unzip.h

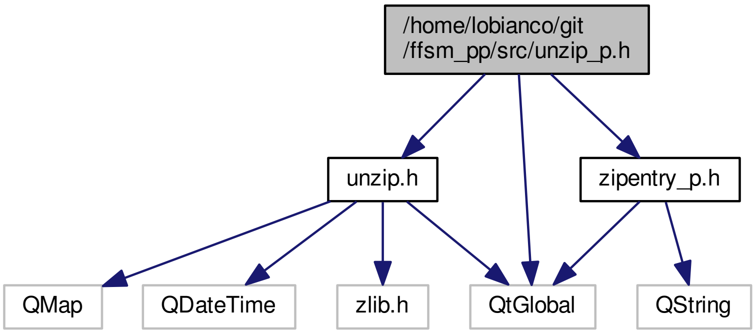
```
/****************************************************************************
** Filename: unzip.h
** Last updated [dd/mm/yyyy]: 28/01/2007
**
** pkzip 2.0 decompression.
**
** Some of the code has been inspired by other open source projects,
** (mainly Info-Zip and Gilles Vollant's minizip).
** Compression and decompression actually uses the zlib library.
**
** Copyright (C) 2007 Angius Fabrizio. All rights reserved.
**
** This file is part of the OSDaB project (http://osdab.sourceforge.net/).
**
** This file may be distributed and/or modified under the terms of the
** GNU General Public License version 2 as published by the Free Software
** Foundation and appearing in the file LICENSE.GPL included in the
** packaging of this file.
**
** This file is provided AS IS with NO WARRANTY OF ANY KIND, INCLUDING THE
** WARRANTY OF DESIGN, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
**
** See the file LICENSE.GPL that came with this software distribution or
** visit http://www.gnu.org/copyleft/gpl.html for GPL licensing information.
**
**********************************************************************/

#ifndef OSDAB_UNZIP__H
#define OSDAB_UNZIP__H

#include <QtGlobal>
#include <QMap>
#include <QDateTime>

#include <zlib.h>

class UnzipPrivate;
class QIODevice;
class QFile;
class QDir;
class QStringList;
class QString;


class UnZip
{
public:
  enum ErrorCode
  {
    Ok,
    ZlibInit,
    ZlibError,
    OpenFailed,
    PartiallyCorrupted,
    Corrupted,
    WrongPassword,
    NoOpenArchive,
    FileNotFound,
    ReadFailed,
    WriteFailed,
    SeekFailed,
    CreateDirFailed,
    InvalidDevice,
    InvalidArchive,
    HeaderConsistencyError,

    Skip, SkipAll // internal use only
  };

  enum ExtractionOption
  {
    //! Extracts paths (default)
    ExtractPaths = 0x0001,
    //! Ignores paths and extracts all the files to the same directory
    SkipPaths = 0x0002
  };
  Q_DECLARE_FLAGS(ExtractionOptions, ExtractionOption)

  enum CompressionMethod
  {
    NoCompression, Deflated, UnknownCompression
  };

  enum FileType
  {
    File, Directory
  };

  typedef struct ZipEntry
  {
    ZipEntry();

    QString filename;
    QString comment;

    quint32 compressedSize;
    quint32 uncompressedSize;
    quint32 crc32;

    QDateTime lastModified;

    CompressionMethod compression;
    FileType type;

    bool encrypted;
  };

  UnZip();
  virtual ~UnZip();

  bool isOpen() const;

  ErrorCode openArchive(const QString& filename);
  ErrorCode openArchive(QIODevice* device);
  void closeArchive();

  QString archiveComment() const;

  QString formatError(UnZip::ErrorCode c) const;

  bool contains(const QString& file) const;

  QStringList fileList() const;
  QList<ZipEntry> entryList() const;

  ErrorCode extractAll(const QString& dirname, ExtractionOptions options = ExtractPaths);
  ErrorCode extractAll(const QDir& dir, ExtractionOptions options = ExtractPaths);

  ErrorCode extractFile(const QString& filename, const QString& dirname, ExtractionOptions options = ExtractPaths);
  ErrorCode extractFile(const QString& filename, const QDir& dir, ExtractionOptions options = ExtractPaths);
  ErrorCode extractFile(const QString& filename, QIODevice* device, ExtractionOptions options = ExtractPaths);

  ErrorCode extractFiles(const QStringList& filenames, const QString& dirname, ExtractionOptions options = ExtractPaths);
  ErrorCode extractFiles(const QStringList& filenames, const QDir& dir, ExtractionOptions options = ExtractPaths);

  void setPassword(const QString& pwd);

private:
  UnzipPrivate* d;
};

Q_DECLARE_OPERATORS_FOR_FLAGS(UnZip::ExtractionOptions)

#endif // OSDAB_UNZIP__H
```

5.147 /home/lobianco/git/ffsm_pp/src/unzip_p.h File Reference

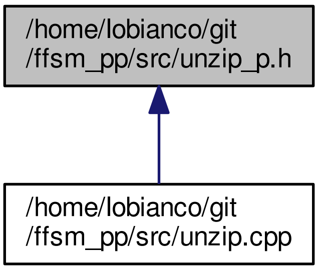
```
#include "unzip.h"
```

```
#include "zipentry_p.h"
#include <QtGlobal>
```

Include dependency graph for unzip_p.h:



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



Classes

- class UnzipPrivate

Macros

- #define UNZIP_READ_BUFFER (256∗1024)

5.147.1 Macro Definition Documentation

5.147.1.1 #define UNZIP_READ_BUFFER (256∗1024)

Definition at line 49 of file unzip_p.h.

Referenced by UnzipPrivate::extractFile().

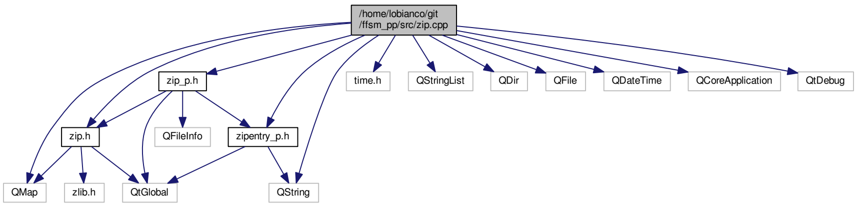
5.148 unzip_p.h

```
/***************************************************************************
** Filename: unzip_p.h
** Last updated [dd/mm/yyyy]: 28/01/2007
**
** pkzip 2.0 decompression.
**
** Some of the code has been inspired by other open source projects,
** (mainly Info-Zip and Gilles Vollant's minizip).
** Compression and decompression actually uses the zlib library.
**
** Copyright (C) 2007 Angius Fabrizio. All rights reserved.
**
** This file is part of the OSDaB project (http://osdab.sourceforge.net/).
**
** This file may be distributed and/or modified under the terms of the
** GNU General Public License version 2 as published by the Free Software
** Foundation and appearing in the file LICENSE.GPL included in the
** packaging of this file.
**
** This file is provided AS IS with NO WARRANTY OF ANY KIND, INCLUDING THE
** WARRANTY OF DESIGN, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
**
** See the file LICENSE.GPL that came with this software distribution or
** visit http://www.gnu.org/copyleft/gpl.html for GPL licensing information.
**
*********************************************************************/

//
//  W A R N I N G
//  -------------
//
// This file is not part of the Zip/UnZip API.  It exists purely as an
// implementation detail. This header file may change from version to
// version without notice, or even be removed.
//
// We mean it.
//

#ifndef OSDAB_UNZIP_P__H
#define OSDAB_UNZIP_P__H

#include "unzip.h"
#include "zipentry_p.h"

#include <QtGlobal>

// zLib authors suggest using larger buffers (128K or 256K) for (de)compression (especially for inflate())
// we use a 256K buffer here - if you want to use this code on a pre-iceage mainframe please change it ;)
#define UNZIP_READ_BUFFER (256*1024)

class UnzipPrivate
{
public:
  UnzipPrivate();

  // Replace this with whatever else you use to store/retrieve the password.
  QString password;

  bool skipAllEncrypted;

  QMap<QString,ZipEntryP*>* headers;

  QIODevice* device;

  char buffer1[UNZIP_READ_BUFFER];
  char buffer2[UNZIP_READ_BUFFER];

  unsigned char* uBuffer;
  const quint32* crcTable;

  // Central Directory (CD) offset
  quint32 cdOffset;
  // End of Central Directory (EOCD) offset
  quint32 eocdOffset;

  // Number of entries in the Central Directory (as to the EOCD record)
  quint16 cdEntryCount;

  // The number of detected entries that have been skipped because of a non compatible format
  quint16 unsupportedEntryCount;

  QString comment;

  UnZip::ErrorCode openArchive(QIODevice* device);

  UnZip::ErrorCode seekToCentralDirectory();
  UnZip::ErrorCode parseCentralDirectoryRecord();
  UnZip::ErrorCode parseLocalHeaderRecord(const QString& path,
ZipEntryP& entry);

  void closeArchive();

  UnZip::ErrorCode extractFile(const QString& path,
ZipEntryP& entry, const QDir& dir, UnZip::ExtractionOptions options);
  UnZip::ErrorCode extractFile(const QString& path,
ZipEntryP& entry, QIODevice* device, UnZip::ExtractionOptions options);

  UnZip::ErrorCode testPassword(quint32* keys, const QString& file, const
ZipEntryP& header);
  bool testKeys(const ZipEntryP& header, quint32* keys);

  bool createDirectory(const QString& path);

  inline void decryptBytes(quint32* keys, char* buffer, qint64 read);

  inline quint32 getULong(const unsigned char* data, quint32 offset) const;
  inline quint64 getULLong(const unsigned char* data, quint32 offset) const;
  inline quint16 getUShort(const unsigned char* data, quint32 offset) const;
  inline int decryptByte(quint32 key2) const;
  inline void updateKeys(quint32* keys, int c) const;
  inline void initKeys(const QString& pwd, quint32* keys) const;

  inline QDateTime convertDateTime(const unsigned char date[2], const unsigned char time[2])
 const;
};

#endif // OSDAB_UNZIP_P__H
```

5.149 /home/lobianco/git/ffsm_pp/src/zip.cpp File Reference

```
#include "zip.h"
#include "zip_p.h"
#include "zipentry_p.h"
#include <time.h>
#include <QMap>
#include <QString>
#include <QStringList>
#include <QDir>
#include <QFile>
#include <QDateTime>
#include <QCoreApplication>
#include <QtDebug>
```

Include dependency graph for zip.cpp:



Macros

- #define ZIP_LOCAL_HEADER_SIZE 30

  Local header size (including signature, excluding variable length fields)
- #define ZIP_LOCAL_ENC_HEADER_SIZE 12

Encryption header size.

- #define ZIP_DD_SIZE_WS 16

  Data descriptor size (signature included)

- #define ZIP_CD_SIZE 46

  Central Directory record size (signature included)

- #define ZIP_EOCD_SIZE 22

  End of Central Directory record size (signature included)

- #define ZIP_LH_OFF_VERS 4
- #define ZIP_LH_OFF_GPFLAG 6
- #define ZIP_LH_OFF_CMET 8
- #define ZIP_LH_OFF_MODT 10
- #define ZIP_LH_OFF_MODD 12
- #define ZIP_LH_OFF_CRC 14
- #define ZIP_LH_OFF_CSIZE 18
- #define ZIP_LH_OFF_USIZE 22
- #define ZIP_LH_OFF_NAMELEN 26
- #define ZIP_LH_OFF_XLEN 28
- #define ZIP_DD_OFF_CRC32 4
- #define ZIP_DD_OFF_CSIZE 8
- #define ZIP_DD_OFF_USIZE 12
- #define ZIP_CD_OFF_MADEBY 4
- #define ZIP_CD_OFF_VERSION 6
- #define ZIP_CD_OFF_GPFLAG 8
- #define ZIP_CD_OFF_CMET 10
- #define ZIP_CD_OFF_MODT 12
- #define ZIP_CD_OFF_MODD 14
- #define ZIP_CD_OFF_CRC 16
- #define ZIP_CD_OFF_CSIZE 20
- #define ZIP_CD_OFF_USIZE 24
- #define ZIP_CD_OFF_NAMELEN 28
- #define ZIP_CD_OFF_XLEN 30
- #define ZIP_CD_OFF_COMMLEN 32
- #define ZIP_CD_OFF_DISKSTART 34
- #define ZIP_CD_OFF_IATTR 36
- #define ZIP_CD_OFF_EATTR 38
- #define ZIP_CD_OFF_LHOFF 42
- #define ZIP_EOCD_OFF_DISKNUM 4
- #define ZIP_EOCD_OFF_CDDISKNUM 6
- #define ZIP_EOCD_OFF_ENTRIES 8
- #define ZIP_EOCD_OFF_CDENTRIES 10
- #define ZIP_EOCD_OFF_CDSIZE 12
- #define ZIP_EOCD_OFF_CDOFF 16
- #define ZIP_EOCD_OFF_COMMLEN 20
- #define ZIP_VERSION 0x14

  PKZip version for archives created by this API.

- #define ZIP_COMPRESSION_THRESHOLD 60

  Do not store very small files as the compression headers overhead would be to big.

- #define CRC32(c, b) crcTable[((int)c^b) & 0xff] ^ (c >> 8)

  This macro updates a one-char-only CRC; it's the Info-Zip macro re-adapted.

5.149.1 Macro Definition Documentation

5.149.1.1 #define CRC32(*c*, *b*) crcTable[((int)c^b) & 0xff] ^ (c >> 8)

This macro updates a one-char-only CRC; it's the Info-Zip macro re-adapted.

Definition at line 108 of file zip.cpp.

Referenced by ZipPrivate::updateKeys().

5.149.1.2 #define ZIP_CD_OFF_CMET 10

Definition at line 78 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.3 #define ZIP_CD_OFF_COMMLEN 32

Definition at line 86 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.4 #define ZIP_CD_OFF_CRC 16

Definition at line 81 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.5 #define ZIP_CD_OFF_CSIZE 20

Definition at line 82 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.6 #define ZIP_CD_OFF_DISKSTART 34

Definition at line 87 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.7 #define ZIP_CD_OFF_EATTR 38

Definition at line 89 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.8 #define ZIP_CD_OFF_GPFLAG 8

Definition at line 77 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.9 #define ZIP_CD_OFF_IATTR 36

Definition at line 88 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.10 #define ZIP_CD_OFF_LHOFF 42

Definition at line 90 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.11 #define ZIP_CD_OFF_MADEBY 4

Definition at line 75 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.12 #define ZIP_CD_OFF_MODD 14

Definition at line 80 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.13 #define ZIP_CD_OFF_MODT 12

Definition at line 79 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.14 #define ZIP_CD_OFF_NAMELEN 28

Definition at line 84 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.15 #define ZIP_CD_OFF_USIZE 24

Definition at line 83 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.16 #define ZIP_CD_OFF_VERSION 6

Definition at line 76 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.17 #define ZIP_CD_OFF_XLEN 30

Definition at line 85 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.18 #define ZIP_CD_SIZE 46

Central Directory record size (signature included)

Definition at line 53 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.19 #define ZIP_COMPRESSION_THRESHOLD 60

Do not store very small files as the compression headers overhead would be to big.

Definition at line 105 of file zip.cpp.

Referenced by ZipPrivate::createEntry().

5.149.1.20 #define ZIP_DD_OFF_CRC32 4

Definition at line 70 of file zip.cpp.

Referenced by ZipPrivate::createEntry().

5.149.1.21 #define ZIP_DD_OFF_CSIZE 8

Definition at line 71 of file zip.cpp.

Referenced by ZipPrivate::createEntry().

5.149.1.22 #define ZIP_DD_OFF_USIZE 12

Definition at line 72 of file zip.cpp.

Referenced by ZipPrivate::createEntry().

5.149.1.23 #define ZIP_DD_SIZE_WS 16

Data descriptor size (signature included)

Definition at line 51 of file zip.cpp.

Referenced by ZipPrivate::createEntry().

5.149.1.24 #define ZIP_EOCD_OFF_CDDISKNUM 6

Definition at line 94 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.25 #define ZIP_EOCD_OFF_CDENTRIES 10

Definition at line 96 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.26 #define ZIP_EOCD_OFF_CDOFF 16

Definition at line 98 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.27 #define ZIP_EOCD_OFF_CDSIZE 12

Definition at line 97 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.28 #define ZIP_EOCD_OFF_COMMLEN 20

Definition at line 99 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.29 #define ZIP_EOCD_OFF_DISKNUM 4

Definition at line 93 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.30 #define ZIP_EOCD_OFF_ENTRIES 8

Definition at line 95 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.31 #define ZIP_EOCD_SIZE 22

End of Central Directory record size (signature included)

Definition at line 55 of file zip.cpp.

Referenced by ZipPrivate::closeArchive().

5.149.1.32 #define ZIP_LH_OFF_CMET 8

Definition at line 60 of file zip.cpp.

Referenced by ZipPrivate::createEntry().

5.149.1.33 #define ZIP_LH_OFF_CRC 14

Definition at line 63 of file zip.cpp.

Referenced by ZipPrivate::createEntry().

5.149.1.34 #define ZIP_LH_OFF_CSIZE 18

Definition at line 64 of file zip.cpp.

Referenced by ZipPrivate::createEntry().

5.149.1.35 #define ZIP_LH_OFF_GPFLAG 6

Definition at line 59 of file zip.cpp.

Referenced by ZipPrivate::createEntry().

5.149.1.36 #define ZIP_LH_OFF_MODD 12

Definition at line 62 of file zip.cpp.

Referenced by ZipPrivate::createEntry().

5.149.1.37 #define ZIP_LH_OFF_MODT 10

Definition at line 61 of file zip.cpp.

Referenced by ZipPrivate::createEntry().

5.149.1.38 #define ZIP_LH_OFF_NAMELEN 26

Definition at line 66 of file zip.cpp.

Referenced by ZipPrivate::createEntry().

5.149.1.39 #define ZIP_LH_OFF_USIZE 22

Definition at line 65 of file zip.cpp.

Referenced by ZipPrivate::createEntry().

5.149.1.40 #define ZIP_LH_OFF_VERS 4

Definition at line 58 of file zip.cpp.

Referenced by ZipPrivate::createEntry().

5.149.1.41 #define ZIP_LH_OFF_XLEN 28

Definition at line 67 of file zip.cpp.

Referenced by ZipPrivate::createEntry().

5.149.1.42 #define ZIP_LOCAL_ENC_HEADER_SIZE 12

Encryption header size.

Definition at line 49 of file zip.cpp.

Referenced by ZipPrivate::createEntry().

5.149.1.43 #define ZIP_LOCAL_HEADER_SIZE 30

Local header size (including signature, excluding variable length fields)

Definition at line 47 of file zip.cpp.

Referenced by ZipPrivate::createEntry().

5.149.1.44 #define ZIP_VERSION 0x14

PKZip version for archives created by this API.

Definition at line 102 of file zip.cpp.

Referenced by ZipPrivate::closeArchive(), and ZipPrivate::createEntry().

5.150 zip.cpp

```
00001 /****************************************************************************
00002 ** Filename: zip.cpp
00003 ** Last updated [dd/mm/yyyy]: 01/02/2007
00004 **
00005 ** pkzip 2.0 file compression.
00006 **
00007 ** Some of the code has been inspired by other open source projects,
00008 ** (mainly Info-Zip and Gilles Vollant's minizip).
00009 ** Compression and decompression actually uses the zlib library.
00010 **
00011 ** Copyright (C) 2007 Angius Fabrizio. All rights reserved.
00012 **
00013 ** This file is part of the OSDaB project (http://osdab.sourceforge.net/).
00014 **
00015 ** This file may be distributed and/or modified under the terms of the
00016 ** GNU General Public License version 2 as published by the Free Software
00017 ** Foundation and appearing in the file LICENSE.GPL included in the
00018 ** packaging of this file.
00019 **
00020 ** This file is provided AS IS with NO WARRANTY OF ANY KIND, INCLUDING THE
00021 ** WARRANTY OF DESIGN, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
00022 **
00023 ** See the file LICENSE.GPL that came with this software distribution or
00024 ** visit http://www.gnu.org/copyleft/gpl.html for GPL licensing information.
00025 **
00026 **********************************************************************/
00027
00028 #include "zip.h"
00029 #include "zip_p.h"
00030 #include "zipentry_p.h"
00031
00032 // we only use this to seed the random number generator
00033 #include <time.h>
00034
00035 #include <QMap>
00036 #include <QString>
00037 #include <QStringList>
00038 #include <QDir>
00039 #include <QFile>
00040 #include <QDateTime>
00041 #include <QCoreApplication>
00042
00043 // You can remove this #include if you replace the qDebug() statements.
00044 #include <QtDebug>
```

```

//! Local header size (including signature, excluding variable length fields)
#define ZIP_LOCAL_HEADER_SIZE 30
//! Encryption header size
#define ZIP_LOCAL_ENC_HEADER_SIZE 12
//! Data descriptor size (signature included)
#define ZIP_DD_SIZE_WS 16
//! Central Directory record size (signature included)
#define ZIP_CD_SIZE 46
//! End of Central Directory record size (signature included)
#define ZIP_EOCD_SIZE 22

// Some offsets inside a local header record (signature included)
#define ZIP_LH_OFF_VERS 4
#define ZIP_LH_OFF_GPFLAG 6
#define ZIP_LH_OFF_CMET 8
#define ZIP_LH_OFF_MODT 10
#define ZIP_LH_OFF_MODD 12
#define ZIP_LH_OFF_CRC 14
#define ZIP_LH_OFF_CSIZE 18
#define ZIP_LH_OFF_USIZE 22
#define ZIP_LH_OFF_NAMELEN 26
#define ZIP_LH_OFF_XLEN 28

// Some offsets inside a data descriptor record (including signature)
#define ZIP_DD_OFF_CRC32 4
#define ZIP_DD_OFF_CSIZE 8
#define ZIP_DD_OFF_USIZE 12

// Some offsets inside a Central Directory record (including signature)
#define ZIP_CD_OFF_MADEBY 4
#define ZIP_CD_OFF_VERSION 6
#define ZIP_CD_OFF_GPFLAG 8
#define ZIP_CD_OFF_CMET 10
#define ZIP_CD_OFF_MODT 12
#define ZIP_CD_OFF_MODD 14
#define ZIP_CD_OFF_CRC 16
#define ZIP_CD_OFF_CSIZE 20
#define ZIP_CD_OFF_USIZE 24
#define ZIP_CD_OFF_NAMELEN 28
#define ZIP_CD_OFF_XLEN 30
#define ZIP_CD_OFF_COMMLEN 32
#define ZIP_CD_OFF_DISKSTART 34
#define ZIP_CD_OFF_IATTR 36
#define ZIP_CD_OFF_EATTR 38
#define ZIP_CD_OFF_LHOFF 42

// Some offsets inside a EOCD record (including signature)
#define ZIP_EOCD_OFF_DISKNUM 4
#define ZIP_EOCD_OFF_CDDISKNUM 6
#define ZIP_EOCD_OFF_ENTRIES 8
#define ZIP_EOCD_OFF_CDENTRIES 10
#define ZIP_EOCD_OFF_CDSIZE 12
#define ZIP_EOCD_OFF_CDOFF 16
#define ZIP_EOCD_OFF_COMMLEN 20

//! PKZip version for archives created by this API
#define ZIP_VERSION 0x14

//! Do not store very small files as the compression headers overhead would be to big
#define ZIP_COMPRESSION_THRESHOLD 60

//! This macro updates a one-char-only CRC; it's the Info-Zip macro re-adapted
#define CRC32(c, b) crcTable[((int)c^b) & 0xff] ^ (c >> 8)

/*!
  \class Zip zip.h

  \brief Zip file compression.

  Some quick usage examples.

  \verbatim
  Suppose you have this directory structure:

  /root/dir1/
  /root/dir1/file1.1
  /root/dir1/file1.2
  /root/dir1/dir1.1/
  /root/dir1/dir1.2/file1.2.1

  EXAMPLE 1:
  myZipInstance.addDirectory("/root/dir1");

  RESULT:
  Beheaves like any common zip software and creates a zip file with this structure:

```

```
  dir1/
  dir1/file1.1
  dir1/file1.2
  dir1/dir1.1/
  dir1/dir1.2/file1.2.1

  EXAMPLE 2:
  myZipInstance.addDirectory("/root/dir1", "myRoot/myFolder");

  RESULT:
  Adds a custom root to the paths and creates a zip file with this structure:

  myRoot/myFolder/dir1/
  myRoot/myFolder/dir1/file1.1
  myRoot/myFolder/dir1/file1.2
  myRoot/myFolder/dir1/dir1.1/
  myRoot/myFolder/dir1/dir1.2/file1.2.1

  EXAMPLE 3:
  myZipInstance.addDirectory("/root/dir1", Zip::AbsolutePaths);

  NOTE:
  Same as calling addDirectory(SOME_PATH, PARENT_PATH_of_SOME_PATH).

  RESULT:
  Preserves absolute paths and creates a zip file with this structure:

  /root/dir1/
  /root/dir1/file1.1
  /root/dir1/file1.2
  /root/dir1/dir1.1/
  /root/dir1/dir1.2/file1.2.1

  EXAMPLE 4:
  myZipInstance.setPassword("hellopass");
  myZipInstance.addDirectory("/root/dir1", "/");

  RESULT:
  Adds and encrypts the files in /root/dir1, creating the following zip structure:

  /dir1/
  /dir1/file1.1
  /dir1/file1.2
  /dir1/dir1.1/
  /dir1/dir1.2/file1.2.1

  \endverbatim
*/

/*! \enum Zip::ErrorCode The result of a compression operation.
  \value Zip::Ok No error occurred.
  \value Zip::ZlibInit Failed to init or load the zlib library.
  \value Zip::ZlibError The zlib library returned some error.
  \value Zip::FileExists The file already exists and will not be overwritten.
  \value Zip::OpenFailed Unable to create or open a device.
  \value Zip::NoOpenArchive CreateArchive() has not been called yet.
  \value Zip::FileNotFound File or directory does not exist.
  \value Zip::ReadFailed Reading of a file failed.
  \value Zip::WriteFailed Writing of a file failed.
  \value Zip::SeekFailed Seek failed.
*/

/*! \enum Zip::CompressionLevel Returns the result of a decompression operation.
  \value Zip::Store No compression.
  \value Zip::Deflate1 Deflate compression level 1(lowest compression).
  \value Zip::Deflate1 Deflate compression level 2.
  \value Zip::Deflate1 Deflate compression level 3.
  \value Zip::Deflate1 Deflate compression level 4.
  \value Zip::Deflate1 Deflate compression level 5.
  \value Zip::Deflate1 Deflate compression level 6.
  \value Zip::Deflate1 Deflate compression level 7.
  \value Zip::Deflate1 Deflate compression level 8.
  \value Zip::Deflate1 Deflate compression level 9 (maximum compression).
  \value Zip::AutoCPU Adapt compression level to CPU speed (faster CPU => better compression).
  \value Zip::AutoMIME Adapt compression level to MIME type of the file being compressed.
  \value Zip::AutoFull Use both CPU and MIME type detection.
*/


/************************************************************************
 Public interface
*************************************************************************/

/*!
  Creates a new Zip file compressor.
*/
Zip::Zip()
```

```
{
  d = new ZipPrivate;
}

/*!
  Closes any open archive and releases used resources.
*/
Zip::~Zip()
{
  closeArchive();
  delete d;
}

/*!
  Returns true if there is an open archive.
*/
bool Zip::isOpen() const
{
  return d->device != 0;
}

/*!
  Sets the password to be used for the next files being added!
  Files added before calling this method will use the previously
  set password (if any).
  Closing the archive won't clear the password!
*/
void Zip::setPassword(const QString& pwd)
{
  d->password = pwd;
}

//! Convenience method, clears the current password.
void Zip::clearPassword()
{
  d->password.clear();
}

//! Returns the currently used password.
QString Zip::password() const
{
  return d->password;
}

/*!
  Attempts to create a new Zip archive. If \p overwrite is true and the file
  already exist it will be overwritten.
  Any open archive will be closed.
 */
Zip::ErrorCode Zip::createArchive(const QString& filename, bool overwrite)
{
  QFile* file = new QFile(filename);

  if (file->exists() && !overwrite) {
    delete file;
    return Zip::FileExists;
  }

  if (!file->open(QIODevice::WriteOnly)) {
    delete file;
    return Zip::OpenFailed;
  }

  Zip::ErrorCode ec = createArchive(file);
  if (ec != Zip::Ok) {
    file->remove();
  }

  return ec;
}

/*!
  Attempts to create a new Zip archive. If there is another open archive this will be closed.
  \warning The class takes ownership of the device!
 */
Zip::ErrorCode Zip::createArchive(QIODevice* device)
{
  if (device == 0)
  {
    qDebug() << "Invalid device.";
    return Zip::OpenFailed;
  }

  return d->createArchive(device);
}

/*!
```

```
  Returns the current archive comment.
*/
QString Zip::archiveComment() const
{
  return d->comment;
}

/*!
  Sets the comment for this archive. Note: createArchive() should have been
  called before.
*/
void Zip::setArchiveComment(const QString& comment)
{
  if (d->device != 0)
    d->comment = comment;
}

/*!
  Convenience method, same as calling
  Zip::addDirectory(const QString&,const QString&,CompressionLevel)
  with an empty \p root parameter (or with the parent directory of \p path if the
  AbsolutePaths options is set).

  The ExtractionOptions are checked in the order they are defined in the zip.h heaser file.
  This means that the last one overwrites the previous one (if some conflict occurs), i.e.
  Zip::IgnorePaths | Zip::AbsolutePaths would be interpreted as Zip::IgnorePaths.
 */
Zip::ErrorCode Zip::addDirectory(const QString& path, CompressionOptions options, CompressionLevel level)
{
  return addDirectory(path, QString(), options, level);
}

/*!
  Convenience method, same as calling Zip::addDirectory(const QString&,const QString&,CompressionOptions,CompressionLevel)
  with the Zip::RelativePaths flag as compression option.
 */
Zip::ErrorCode Zip::addDirectory(const QString& path, const QString& root, CompressionLevel level)
{
  return addDirectory(path, root, Zip::RelativePaths, level);
}

/*!
  Convenience method, same as calling Zip::addDirectory(const QString&,const QString&,CompressionOptions,CompressionLevel)
  with the Zip::IgnorePaths flag as compression option and an empty \p root parameter.
*/
Zip::ErrorCode Zip::addDirectoryContents(const QString& path, CompressionLevel level)
{
  return addDirectory(path, QString(), IgnorePaths, level);
}

/*!
  Convenience method, same as calling Zip::addDirectory(const QString&,const QString&,CompressionOptions,CompressionLevel)
  with the Zip::IgnorePaths flag as compression option.
*/
Zip::ErrorCode Zip::addDirectoryContents(const QString& path, const QString& root, CompressionLevel level)
{
  return addDirectory(path, root, IgnorePaths, level);
}

/*!
  Recursively adds files contained in \p dir to the archive, using \p root as name for the root folder.
  Stops adding files if some error occurs.

  The ExtractionOptions are checked in the order they are defined in the zip.h heaser file.
  This means that the last one overwrites the previous one (if some conflict occurs), i.e.
  Zip::IgnorePaths | Zip::AbsolutePaths would be interpreted as Zip::IgnorePaths.

  The \p root parameter is ignored with the Zip::IgnorePaths parameter and used as path prefix (a trailing /
  is always added as directory separator!) otherwise (even with Zip::AbsolutePaths set!).
*/
Zip::ErrorCode Zip::addDirectory(const QString& path, const QString& root, CompressionOptions options, CompressionLevel level)
{
  // qDebug() << QString("addDir(path=%1, root=%2)").arg(path, root);

  // Bad boy didn't call createArchive() yet :)
  if (d->device == 0)
    return Zip::NoOpenArchive;

  QDir dir(path);
  if (!dir.exists())
    return Zip::FileNotFound;

  // Remove any trailing separator
  QString actualRoot = root.trimmed();

  // Preserve Unix root
  if (actualRoot != "/")
  {
    while (actualRoot.endsWith("/") || actualRoot.endsWith("\\"))
      actualRoot.truncate(actualRoot.length() - 1);
  }

  // QDir::cleanPath() fixes some issues with QDir::dirName()
  QFileInfo current(QDir::cleanPath(path));

  if (!actualRoot.isEmpty() && actualRoot != "/")
    actualRoot.append("/");

  /* This part is quite confusing and needs some test or check */
  /* An attempt to compress the / root directory evtl. using a root prefix should be a good test */
  if (options.testFlag(AbsolutePaths) && !options.testFlag(IgnorePaths))
  {
    QString absolutePath = d->extractRoot(path);
    if (!absolutePath.isEmpty() && absolutePath != "/")
      absolutePath.append("/");
    actualRoot.append(absolutePath);
  }

  if (!options.testFlag(IgnorePaths))
  {
    actualRoot = actualRoot.append(QDir(current.absoluteFilePath()).dirName());
    actualRoot.append("/");
  }

  // actualRoot now contains the path of the file relative to the zip archive
  // with a trailing /

  QFileInfoList list = dir.entryInfoList(
    QDir::Files |
    QDir::Dirs |
    QDir::NoDotAndDotDot |
    QDir::NoSymLinks);

  ErrorCode ec = Zip::Ok;
  bool filesAdded = false;

  CompressionOptions recursionOptions;
  if (options.testFlag(IgnorePaths))
    recursionOptions |= IgnorePaths;
  else recursionOptions |= RelativePaths;

  for (int i = 0; i < list.size() && ec == Zip::Ok; ++i)
  {
    QFileInfo info = list.at(i);

    if (info.isDir())
    {
      // Recursion :)
      ec = addDirectory(info.absoluteFilePath(), actualRoot, recursionOptions, level);
    }
    else
    {
      ec = d->createEntry(info, actualRoot, level);
      filesAdded = true;
    }
  }


  // We need an explicit record for this dir
  // Non-empty directories don't need it because they have a path component in the filename
  if (!filesAdded && !options.testFlag(IgnorePaths))
    ec = d->createEntry(current, actualRoot, level);

  return ec;
}

/*!
  Closes the archive and writes any pending data.
*/
Zip::ErrorCode Zip::closeArchive()
{
  Zip::ErrorCode ec = d->closeArchive();
  d->reset();
  return ec;
```

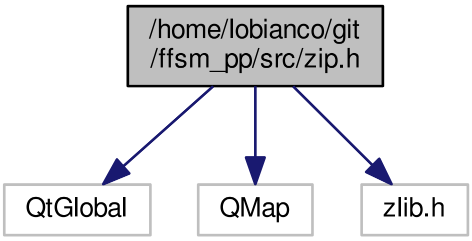
```
}

/*!
  Returns a locale translated error string for a given error code.
*/
QString Zip::formatError(Zip::ErrorCode c) const
{
  switch (c)
  {
  case Ok: return QCoreApplication::translate("Zip", "ZIP operation completed successfully."); break;
  case ZlibInit: return QCoreApplication::translate("Zip", "Failed to initialize or load zlib library."); break;
  case ZlibError: return QCoreApplication::translate("Zip", "zlib library error."); break;
  case OpenFailed: return QCoreApplication::translate("Zip", "Unable to create or open file."); break;
  case NoOpenArchive: return QCoreApplication::translate("Zip", "No archive has been created yet."); break;
  case FileNotFound: return QCoreApplication::translate("Zip", "File or directory does not exist."); break;
  case ReadFailed: return QCoreApplication::translate("Zip", "File read error."); break;
  case WriteFailed: return QCoreApplication::translate("Zip", "File write error."); break;
  case SeekFailed: return QCoreApplication::translate("Zip", "File seek error."); break;
  default: ;
  }

  return QCoreApplication::translate("Zip", "Unknown error.");
}


/************************************************************************
 Private interface
*************************************************************************/

//! \internal
ZipPrivate::ZipPrivate()
{
  headers = 0;
  device = 0;

  // keep an unsigned pointer so we avoid to over bloat the code with casts
  uBuffer = (unsigned char*) buffer1;
  crcTable = (quint32*) get_crc_table();
}

//! \internal
ZipPrivate::~ZipPrivate()
{
  closeArchive();
}

//! \internal
Zip::ErrorCode ZipPrivate::createArchive(QIODevice* dev)
{
  Q_ASSERT(dev != 0);

  if (device != 0)
    closeArchive();

  device = dev;

  if (!device->isOpen())
  {
    if (!device->open(QIODevice::ReadOnly)) {
      delete device;
      device = 0;
      qDebug() << "Unable to open device for writing.";
      return Zip::OpenFailed;
    }
  }

  headers = new QMap<QString,ZipEntryP*>;
  return Zip::Ok;
}

//! \internal Writes a new entry in the zip file.
Zip::ErrorCode ZipPrivate::createEntry(const QFileInfo& file, const QString& root, Zip::CompressionLevel level)
{
  //! \todo Automatic level detection (cpu, extension & file size)

  // Directories and very small files are always stored
  // (small files would get bigger due to the compression headers overhead)

  // Need this for zlib
  bool isPNGFile = false;
  bool dirOnly = file.isDir();

  QString entryName = root;

  // Directory entry
  if (dirOnly)
    level = Zip::Store;
  else
  {
    entryName.append(file.fileName());

    QString ext = file.completeSuffix().toLower();
    isPNGFile = ext == "png";

    if (file.size() < ZIP_COMPRESSION_THRESHOLD)
      level = Zip::Store;
    else
      switch (level)
      {
      case Zip::AutoCPU:
        level = Zip::Deflate5;
        break;
      case Zip::AutoMIME:
        level = detectCompressionByMime(ext);
        break;
      case Zip::AutoFull:
        level = detectCompressionByMime(ext);
        break;
      default:
        ;
      }
  }

  // entryName contains the path as it should be written
  // in the zip file records
  // qDebug() << QString("addDir(file=%1, root=%2, entry=%3)").arg(file.absoluteFilePath(), root,
 entryName);

  // create header and store it to write a central directory later
  ZipEntryP* h = new ZipEntryP;

  h->compMethod = (level == Zip::Store) ? 0 : 0x0008;

  // Set encryption bit and set the data descriptor bit
  // so we can use mod time instead of crc for password check
  bool encrypt = !dirOnly && !password.isEmpty();
  if (encrypt)
    h->gpFlag[0] |= 9;

  QDateTime dt = file.lastModified();
  QDate d = dt.date();
  h->modDate[1] = ((d.year() - 1980) << 1) & 254;
  h->modDate[1] |= ((d.month() >> 3) & 1);
  h->modDate[0] = ((d.month() & 7) << 5) & 224;
  h->modDate[0] |= d.day();

  QTime t = dt.time();
  h->modTime[1] = (t.hour() << 3) & 248;
  h->modTime[1] |= ((t.minute() >> 3) & 7);
  h->modTime[0] = ((t.minute() & 7) << 5) & 224;
  h->modTime[0] |= t.second() / 2;

  h->szUncomp = dirOnly ? 0 : file.size();

  // **** Write local file header ****

  // signature
  buffer1[0] = 'P'; buffer1[1] = 'K';
  buffer1[2] = 0x3; buffer1[3] = 0x4;

  // version needed to extract
  buffer1[ZIP_LH_OFF_VERS] = ZIP_VERSION;
  buffer1[ZIP_LH_OFF_VERS + 1] = 0;

  // general purpose flag
  buffer1[ZIP_LH_OFF_GPFLAG] = h->gpFlag[0];
  buffer1[ZIP_LH_OFF_GPFLAG + 1] = h->gpFlag[1];

  // compression method
  buffer1[ZIP_LH_OFF_CMET] = h->compMethod & 0xFF;
  buffer1[ZIP_LH_OFF_CMET + 1] = (h->compMethod>>8) & 0xFF;

  // last mod file time
  buffer1[ZIP_LH_OFF_MODT] = h->modTime[0];
  buffer1[ZIP_LH_OFF_MODT + 1] = h->modTime[1];

  // last mod file date
  buffer1[ZIP_LH_OFF_MODD] = h->modDate[0];
  buffer1[ZIP_LH_OFF_MODD + 1] = h->modDate[1];
```

```

  // skip crc (4bytes) [14,15,16,17]

  // skip compressed size but include evtl. encryption header (4bytes: [18,19,20,21])
  buffer1[ZIP_LH_OFF_CSIZE] =
  buffer1[ZIP_LH_OFF_CSIZE + 1] =
  buffer1[ZIP_LH_OFF_CSIZE + 2] =
  buffer1[ZIP_LH_OFF_CSIZE + 3] = 0;

  h->szComp = encrypt ? ZIP_LOCAL_ENC_HEADER_SIZE : 0;

  // uncompressed size [22,23,24,25]
  setULong(h->szUncomp, buffer1, ZIP_LH_OFF_USIZE);

  // filename length
    //QByteArray entryNameBytes = entryName.toAscii();
    QByteArray entryNameBytes = entryName.toLatin1(); // Qt5
  int sz = entryNameBytes.size();

  buffer1[ZIP_LH_OFF_NAMELEN] = sz & 0xFF;
  buffer1[ZIP_LH_OFF_NAMELEN + 1] = (sz >> 8) & 0xFF;

  // extra field length
  buffer1[ZIP_LH_OFF_XLEN] = buffer1[ZIP_LH_OFF_XLEN + 1] = 0;

  // Store offset to write crc and compressed size
  h->lhOffset = device->pos();
  quint32 crcOffset = h->lhOffset + ZIP_LH_OFF_CRC;

  if (device->write(buffer1, ZIP_LOCAL_HEADER_SIZE) !=
ZIP_LOCAL_HEADER_SIZE)
  {
    delete h;
    return Zip::WriteFailed;
  }

  // Write out filename
  if (device->write(entryNameBytes) != sz)
  {
    delete h;
    return Zip::WriteFailed;
  }

  // Encryption keys
  quint32 keys[3] = { 0, 0, 0 };

  if (encrypt)
  {
    // **** encryption header ****

    // XOR with PI to ensure better random numbers
    // with poorly implemented rand() as suggested by Info-Zip
    srand(time(NULL) ^ 3141592654UL);
    int randByte;

    initKeys(keys);
    for (int i=0; i<10; ++i)
    {
      randByte = (rand() >> 7) & 0xff;
      buffer1[i] = decryptByte(keys[2]) ^ randByte;
      updateKeys(keys, randByte);
    }

    // Encrypt encryption header
    initKeys(keys);
    for (int i=0; i<10; ++i)
    {
      randByte = decryptByte(keys[2]);
      updateKeys(keys, buffer1[i]);
      buffer1[i] ^= randByte;
    }

    // We don't know the CRC at this time, so we use the modification time
    // as the last two bytes
    randByte = decryptByte(keys[2]);
    updateKeys(keys, h->modTime[0]);
    buffer1[10] ^= randByte;

    randByte = decryptByte(keys[2]);
    updateKeys(keys, h->modTime[1]);
    buffer1[11] ^= randByte;

    // Write out encryption header
    if (device->write(buffer1, ZIP_LOCAL_ENC_HEADER_SIZE) !=
ZIP_LOCAL_ENC_HEADER_SIZE)
    {
      delete h;
      return Zip::WriteFailed;
    }
  }

  qint64 written = 0;
  quint32 crc = crc32(0L, Z_NULL, 0);

  if (!dirOnly)
  {
    QFile actualFile(file.absoluteFilePath());
    if (!actualFile.open(QIODevice::ReadOnly))
    {
      qDebug() << QString("An error occurred while opening %1").arg(file.absoluteFilePath());
      return Zip::OpenFailed;
    }

    // Write file data
    qint64 read = 0;
    qint64 totRead = 0;
    qint64 toRead = actualFile.size();

    if (level == Zip::Store)
    {
      while ( (read = actualFile.read(buffer1, ZIP_READ_BUFFER)) > 0 )
      {
        crc = crc32(crc, uBuffer, read);

        if (password != 0)
          encryptBytes(keys, buffer1, read);

        if ( (written = device->write(buffer1, read)) != read )
        {
          actualFile.close();
          delete h;
          return Zip::WriteFailed;
        }
      }
    }
    else
    {
      z_stream zstr;

      // Initialize zalloc, zfree and opaque before calling the init function
      zstr.zalloc = Z_NULL;
      zstr.zfree = Z_NULL;
      zstr.opaque = Z_NULL;

      int zret;

      // Use deflateInit2 with negative windowBits to get raw compression
      if ((zret = deflateInit2_(
          &zstr,
          (int)level,
          Z_DEFLATED,
          -MAX_WBITS,
          8,
          isPNGFile ? Z_RLE : Z_DEFAULT_STRATEGY,
          ZLIB_VERSION,
          sizeof(z_stream)
        )) != Z_OK )
      {
        actualFile.close();
        qDebug() << "Could not initialize zlib for compression";
        delete h;
        return Zip::ZlibError;
      }

      qint64 compressed;

      int flush = Z_NO_FLUSH;

      do
      {
        read = actualFile.read(buffer1, ZIP_READ_BUFFER);
        totRead += read;

        if (read == 0)
          break;
        if (read < 0)
        {
          actualFile.close();
          deflateEnd(&zstr);
          qDebug() << QString("Error while reading %1").arg(file.absoluteFilePath());
          delete h;
          return Zip::ReadFailed;
        }

        crc = crc32(crc, uBuffer, read);

        zstr.next_in = (Bytef*) buffer1;
        zstr.avail_in = (uInt)read;

        // Tell zlib if this is the last chunk we want to encode
        // by setting the flush parameter to Z_FINISH
        flush = (totRead == toRead) ? Z_FINISH : Z_NO_FLUSH;

        // Run deflate() on input until output buffer not full
        // finish compression if all of source has been read in
            do
            {
          zstr.next_out = (Bytef*) buffer2;
          zstr.avail_out = ZIP_READ_BUFFER;

          zret = deflate(&zstr, flush);
          // State not clobbered
          Q_ASSERT(zret != Z_STREAM_ERROR);

          // Write compressed data to file and empty buffer
          compressed = ZIP_READ_BUFFER - zstr.avail_out;

          if (password != 0)
            encryptBytes(keys, buffer2, compressed);

          if (device->write(buffer2, compressed) != compressed)
          {
            deflateEnd(&zstr);
            actualFile.close();
            qDebug() << QString("Error while writing %1").arg(file.absoluteFilePath());
            delete h;
            return Zip::WriteFailed;
          }

          written += compressed;

        } while (zstr.avail_out == 0);

        // All input will be used
        Q_ASSERT(zstr.avail_in == 0);

      } while (flush != Z_FINISH);

      // Stream will be complete
      Q_ASSERT(zret == Z_STREAM_END);

      deflateEnd(&zstr);

    } // if (level != STORE)

    actualFile.close();
  }

  // Store end of entry offset
  quint32 current = device->pos();

  // Update crc and compressed size in local header
  if (!device->seek(crcOffset))
  {
    delete h;
    return Zip::SeekFailed;
  }

  h->crc = dirOnly ? 0 : crc;
  h->szComp += written;

  setULong(h->crc, buffer1, 0);
  setULong(h->szComp, buffer1, 4);
  if ( device->write(buffer1, 8) != 8)
  {
    delete h;
    return Zip::WriteFailed;
  }

  // Seek to end of entry
  if (!device->seek(current))
  {
    delete h;
    return Zip::SeekFailed;
  }

  if ((h->gpFlag[0] & 8) == 8)
  {
    // Write data descriptor

    // Signature: PK\7\8
    buffer1[0] = 'P';
    buffer1[1] = 'K';
    buffer1[2] = 0x07;
    buffer1[3] = 0x08;

    // CRC
    setULong(h->crc, buffer1, ZIP_DD_OFF_CRC32);

    // Compressed size
    setULong(h->szComp, buffer1, ZIP_DD_OFF_CSIZE);

    // Uncompressed size
    setULong(h->szUncomp, buffer1, ZIP_DD_OFF_USIZE);

    if (device->write(buffer1, ZIP_DD_SIZE_WS) != ZIP_DD_SIZE_WS)
    {
      delete h;
      return Zip::WriteFailed;
    }
  }

  headers->insert(entryName, h);
  return Zip::Ok;
}

//! \internal
int ZipPrivate::decryptByte(quint32 key2) const
{
  quint16 temp = ((quint16)(key2) & 0xffff) | 2;
  return (int)(((temp * (temp ^ 1)) >> 8) & 0xff);
}

//! \internal Writes an quint32 (4 bytes) to a byte array at given offset.
void ZipPrivate::setULong(quint32 v, char* buffer, unsigned int offset)
{
  buffer[offset+3] = ((v >> 24) & 0xFF);
  buffer[offset+2] = ((v >> 16) & 0xFF);
  buffer[offset+1] = ((v >> 8) & 0xFF);
  buffer[offset] = (v & 0xFF);
}

//! \internal Initializes decryption keys using a password.
void ZipPrivate::initKeys(quint32* keys) const
{
  // Encryption keys initialization constants are taken from the
  // PKZip file format specification docs
  keys[0] = 305419896L;
  keys[1] = 591751049L;
  keys[2] = 878082192L;

    //QByteArray pwdBytes = password.toAscii();
    QByteArray pwdBytes = password.toLatin1();
  int sz = pwdBytes.size();
  const char* ascii = pwdBytes.data();

  for (int i=0; i<sz; ++i)
    updateKeys(keys, (int)ascii[i]);
}

//! \internal Updates encryption keys.
void ZipPrivate::updateKeys(quint32* keys, int c) const
{
  keys[0] = CRC32(keys[0], c);
  keys[1] += keys[0] & 0xff;
  keys[1] = keys[1] * 134775813L + 1;
  keys[2] = CRC32(keys[2], ((int)keys[1]) >> 24);
}

//! \internal Encrypts a byte array.
void ZipPrivate::encryptBytes(quint32* keys, char* buffer, qint64 read)
{
  char t;

  for (int i=0; i<(int)read; ++i)
  {
    t = buffer[i];
    buffer[i] ^= decryptByte(keys[2]);
    updateKeys(keys, t);
  }
}

//! \internal Detects the best compression level for a given file extension.
Zip::CompressionLevel ZipPrivate::detectCompressionByMime
(const QString& ext)
{
  // files really hard to compress
  if ((ext == "png") ||
    (ext == "jpg") ||
    (ext == "jpeg") ||
    (ext == "mp3") ||
    (ext == "ogg") ||
    (ext == "ogm") ||
    (ext == "avi") ||
    (ext == "mov") ||
    (ext == "rm") ||
    (ext == "ra") ||
    (ext == "zip") ||
    (ext == "rar") ||
    (ext == "bz2") ||
    (ext == "gz") ||
    (ext == "7z") ||
    (ext == "z") ||
    (ext == "jar")
  ) return Zip::Store;

  // files slow and hard to compress
  if ((ext == "exe") ||
    (ext == "bin") ||
    (ext == "rpm") ||
    (ext == "deb")
  ) return Zip::Deflate2;

  return Zip::Deflate9;
}

/*!
  Closes the current archive and writes out pending data.
*/
Zip::ErrorCode ZipPrivate::closeArchive()
{
  // Close current archive by writing out central directory
  // and free up resources

  if (device == 0)
    return Zip::Ok;

  if (headers == 0)
    return Zip::Ok;

  const ZipEntryP* h;

  unsigned int sz;
  quint32 szCentralDir = 0;
  quint32 offCentralDir = device->pos();

  for (QMap<QString,ZipEntryP*>::ConstIterator itr = headers->constBegin(); itr != headers->constEnd(); ++itr)
  {
    h = itr.value();

    // signature
    buffer1[0] = 'P';
    buffer1[1] = 'K';
    buffer1[2] = 0x01;
    buffer1[3] = 0x02;

    // version made by  (currently only MS-DOS/FAT - no symlinks or other stuff supported)
    buffer1[ZIP_CD_OFF_MADEBY] = buffer1[ZIP_CD_OFF_MADEBY + 1] = 0;

    // version needed to extract
    buffer1[ZIP_CD_OFF_VERSION] = ZIP_VERSION;
    buffer1[ZIP_CD_OFF_VERSION + 1] = 0;

    // general purpose flag
    buffer1[ZIP_CD_OFF_GPFLAG] = h->gpFlag[0];
    buffer1[ZIP_CD_OFF_GPFLAG + 1] = h->gpFlag[1];

    // compression method
    buffer1[ZIP_CD_OFF_CMET] = h->compMethod & 0xFF;
    buffer1[ZIP_CD_OFF_CMET + 1] = (h->compMethod >> 8) & 0xFF;

    // last mod file time
    buffer1[ZIP_CD_OFF_MODT] = h->modTime[0];
    buffer1[ZIP_CD_OFF_MODT + 1] = h->modTime[1];

    // last mod file date
    buffer1[ZIP_CD_OFF_MODD] = h->modDate[0];
    buffer1[ZIP_CD_OFF_MODD + 1] = h->modDate[1];

    // crc (4bytes) [16,17,18,19]
    setULong(h->crc, buffer1, ZIP_CD_OFF_CRC);

    // compressed size (4bytes: [20,21,22,23])
    setULong(h->szComp, buffer1, ZIP_CD_OFF_CSIZE);

    // uncompressed size [24,25,26,27]
    setULong(h->szUncomp, buffer1, ZIP_CD_OFF_USIZE);

    // filename
        //QByteArray fileNameBytes = itr.key().toAscii();
        QByteArray fileNameBytes = itr.key().toLatin1();
    sz = fileNameBytes.size();
    buffer1[ZIP_CD_OFF_NAMELEN] = sz & 0xFF;
    buffer1[ZIP_CD_OFF_NAMELEN + 1] = (sz >> 8) & 0xFF;

    // extra field length
    buffer1[ZIP_CD_OFF_XLEN] = buffer1[ZIP_CD_OFF_XLEN + 1] = 0;

    // file comment length
    buffer1[ZIP_CD_OFF_COMMLEN] = buffer1[ZIP_CD_OFF_COMMLEN + 1] = 0;

    // disk number start
    buffer1[ZIP_CD_OFF_DISKSTART] = buffer1[
ZIP_CD_OFF_DISKSTART + 1] = 0;

    // internal file attributes
    buffer1[ZIP_CD_OFF_IATTR] = buffer1[ZIP_CD_OFF_IATTR + 1] = 0;

    // external file attributes
    buffer1[ZIP_CD_OFF_EATTR] =
    buffer1[ZIP_CD_OFF_EATTR + 1] =
    buffer1[ZIP_CD_OFF_EATTR + 2] =
    buffer1[ZIP_CD_OFF_EATTR + 3] = 0;

    // relative offset of local header [42->45]
    setULong(h->lhOffset, buffer1, ZIP_CD_OFF_LHOFF);

    if (device->write(buffer1, ZIP_CD_SIZE) != ZIP_CD_SIZE)
    {
      //! \todo See if we can detect QFile objects using the Qt Meta Object System
      /*
      if (!device->remove())
        qDebug() << tr("Unable to delete corrupted archive: %1").arg(device->fileName());
      */
      return Zip::WriteFailed;
    }

    // Write out filename
    if ((unsigned int)device->write(fileNameBytes) != sz)
    {
      //! \todo SAME AS ABOVE: See if we can detect QFile objects using the Qt Meta Object System
      /*
      if (!device->remove())
        qDebug() << tr("Unable to delete corrupted archive: %1").arg(device->fileName());
        */
      return Zip::WriteFailed;
    }

    szCentralDir += (ZIP_CD_SIZE + sz);

  } // central dir headers loop


  // Write end of central directory

  // signature
  buffer1[0] = 'P';
  buffer1[1] = 'K';
  buffer1[2] = 0x05;
  buffer1[3] = 0x06;

  // number of this disk
  buffer1[ZIP_EOCD_OFF_DISKNUM] = buffer1[
ZIP_EOCD_OFF_DISKNUM + 1] = 0;

  // number of disk with central directory
  buffer1[ZIP_EOCD_OFF_CDDISKNUM] = buffer1[
ZIP_EOCD_OFF_CDDISKNUM + 1] = 0;

  // number of entries in this disk
  sz = headers->count();
    buffer1[ZIP_EOCD_OFF_ENTRIES] = sz & 0xFF;
  buffer1[ZIP_EOCD_OFF_ENTRIES + 1] = (sz >> 8) & 0xFF;

  // total number of entries
  buffer1[ZIP_EOCD_OFF_CDENTRIES] = buffer1[
ZIP_EOCD_OFF_ENTRIES];
  buffer1[ZIP_EOCD_OFF_CDENTRIES + 1] = buffer1[
ZIP_EOCD_OFF_ENTRIES + 1];

  // size of central directory [12->15]
  setULong(szCentralDir, buffer1, ZIP_EOCD_OFF_CDSIZE);

  // central dir offset [16->19]
  setULong(offCentralDir, buffer1, ZIP_EOCD_OFF_CDOFF);

  // ZIP file comment length
    //QByteArray commentBytes = comment.toAscii();
    QByteArray commentBytes = comment.toLatin1();
  quint16 commentLength = commentBytes.size();

  if (commentLength == 0)
  {
    buffer1[ZIP_EOCD_OFF_COMMLEN] = buffer1[
ZIP_EOCD_OFF_COMMLEN + 1] = 0;
  }
  else
  {
    buffer1[ZIP_EOCD_OFF_COMMLEN] = commentLength & 0xFF;
    buffer1[ZIP_EOCD_OFF_COMMLEN + 1] = (commentLength >> 8) & 0xFF;
  }

  if (device->write(buffer1, ZIP_EOCD_SIZE) != ZIP_EOCD_SIZE)
  {
    //! \todo SAME AS ABOVE: See if we can detect QFile objects using the Qt Meta Object System
    /*
    if (!device->remove())
      qDebug() << tr("Unable to delete corrupted archive: %1").arg(device->fileName());
      */
    return Zip::WriteFailed;
  }

  if (commentLength != 0)
  {
    if ((unsigned int)device->write(commentBytes) != commentLength)
    {
      //! \todo SAME AS ABOVE: See if we can detect QFile objects using the Qt Meta Object System
      /*
      if (!device->remove())
        qDebug() << tr("Unable to delete corrupted archive: %1").arg(device->fileName());
        */
      return Zip::WriteFailed;
    }
  }

  return Zip::Ok;
}

//! \internal
void ZipPrivate::reset()
{
  comment.clear();

  if (headers != 0)
  {
    qDeleteAll(*headers);
    delete headers;
    headers = 0;
  }

  delete device; device = 0;
}

//! \internal Returns the path of the parent directory
QString ZipPrivate::extractRoot(const QString& p)
{
  QDir d(QDir::cleanPath(p));
  if (!d.exists())
    return QString();

  if (!d.cdUp())
    return QString();

  return d.absolutePath();
}
```

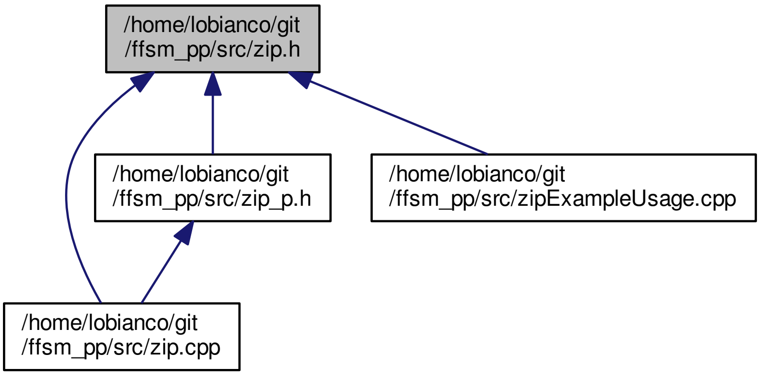
5.151 /home/lobianco/git/ffsm_pp/src/zip.h File Reference

```
#include <QtGlobal>
#include <QMap>
#include <zlib.h>
```

Include dependency graph for zip.h:



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



Classes

- class Zip

  Zip file compression.

5.152 zip.h

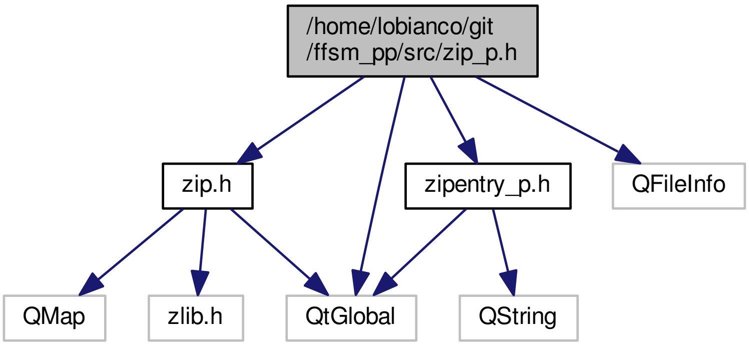
```
/*****************************************************************************
** Filename: zip.h
** Last updated [dd/mm/yyyy]: 01/02/2007
**
** pkzip 2.0 file compression.
**
** Some of the code has been inspired by other open source projects,
** (mainly Info-Zip and Gilles Vollant's minizip).
** Compression and decompression actually uses the zlib library.
**
** Copyright (C) 2007 Angius Fabrizio. All rights reserved.
**
```

```
** This file is part of the OSDaB project (http://osdab.sourceforge.net/).
**
** This file may be distributed and/or modified under the terms of the
** GNU General Public License version 2 as published by the Free Software
** Foundation and appearing in the file LICENSE.GPL included in the
** packaging of this file.
**
** This file is provided AS IS with NO WARRANTY OF ANY KIND, INCLUDING THE
** WARRANTY OF DESIGN, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
**
** See the file LICENSE.GPL that came with this software distribution or
** visit http://www.gnu.org/copyleft/gpl.html for GPL licensing information.
**
**********************************************************************/

#ifndef OSDAB_ZIP__H
#define OSDAB_ZIP__H

#include <QtGlobal>
#include <QMap>

#include <zlib.h>

class ZipPrivate;

class QIODevice;
class QFile;
class QDir;
class QStringList;
class QString;


class Zip
{
public:
  enum ErrorCode
  {
    Ok,
    ZlibInit,
    ZlibError,
    FileExists,
    OpenFailed,
    NoOpenArchive,
    FileNotFound,
    ReadFailed,
    WriteFailed,
    SeekFailed
  };

  enum CompressionLevel
  {
    Store,
    Deflate1 = 1, Deflate2, Deflate3, Deflate4,
    Deflate5, Deflate6, Deflate7, Deflate8,
Deflate9,
    AutoCPU, AutoMIME, AutoFull
  };

  enum CompressionOption
  {
    //! Does not preserve absolute paths in the zip file when adding a file/directory (default)
    RelativePaths = 0x0001,
    //! Preserve absolute paths
    AbsolutePaths = 0x0002,
    //! Do not store paths. All the files are put in the (evtl. user defined) root of the zip file
    IgnorePaths = 0x0004
  };
  Q_DECLARE_FLAGS(CompressionOptions, CompressionOption)

  Zip();
  virtual ~Zip();

  bool isOpen() const;

  void setPassword(const QString& pwd);
  void clearPassword();
  QString password() const;

  ErrorCode createArchive(const QString& file, bool overwrite = true);
  ErrorCode createArchive(QIODevice* device);

  QString archiveComment() const;
  void setArchiveComment(const QString& comment);

  ErrorCode addDirectoryContents(const QString& path,
CompressionLevel level = AutoFull);
  ErrorCode addDirectoryContents(const QString& path, const QString& root,
CompressionLevel level = AutoFull);

  ErrorCode addDirectory(const QString& path, CompressionOptions options =
RelativePaths, CompressionLevel level = AutoFull);
  ErrorCode addDirectory(const QString& path, const QString& root,
CompressionLevel level = AutoFull);
  ErrorCode addDirectory(const QString& path, const QString& root, CompressionOptions
options = RelativePaths, CompressionLevel level = AutoFull);

  ErrorCode closeArchive();

  QString formatError(ErrorCode c) const;

private:
  ZipPrivate* d;
};

Q_DECLARE_OPERATORS_FOR_FLAGS(Zip::CompressionOptions)

#endif // OSDAB_ZIP__H
```

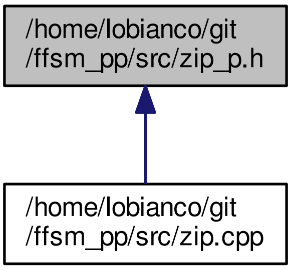
5.153 /home/lobianco/git/ffsm_pp/src/zip_p.h File Reference

```
#include "zip.h"
#include "zipentry_p.h"
#include <QtGlobal>
#include <QFileInfo>
```

Include dependency graph for zip_p.h:



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



Classes

- class ZipPrivate

Macros

- #define ZIP_READ_BUFFER (256∗1024)

5.153.1 Macro Definition Documentation

5.153.1.1 #define ZIP_READ_BUFFER (256∗1024)

zLib authors suggest using larger buffers (128K or 256K) for (de)compression (especially for inflate()) we use a 256K buffer here - if you want to use this code on a pre-iceage mainframe please change it ;)

Definition at line 52 of file zip_p.h.

Referenced by ZipPrivate::createEntry().

5.154 zip_p.h

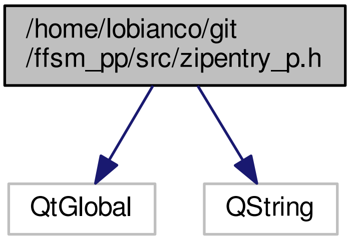
```
/****************************************************************************
** Filename: zip_p.h
** Last updated [dd/mm/yyyy]: 28/01/2007
**
** pkzip 2.0 file compression.
**
** Some of the code has been inspired by other open source projects,
** (mainly Info-Zip and Gilles Vollant’s minizip).
** Compression and decompression actually uses the zlib library.
**
** Copyright (C) 2007 Angius Fabrizio. All rights reserved.
**
** This file is part of the OSDaB project (http://osdab.sourceforge.net/).
**
** This file may be distributed and/or modified under the terms of the
** GNU General Public License version 2 as published by the Free Software
** Foundation and appearing in the file LICENSE.GPL included in the
** packaging of this file.
```

```
**
** This file is provided AS IS with NO WARRANTY OF ANY KIND, INCLUDING THE
** WARRANTY OF DESIGN, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
**
** See the file LICENSE.GPL that came with this software distribution or
** visit http://www.gnu.org/copyleft/gpl.html for GPL licensing information.
**
**********************************************************************/

//
//  W A R N I N G
//  -------------
//
// This file is not part of the Zip/UnZip API.  It exists purely as an
// implementation detail. This header file may change from version to
// version without notice, or even be removed.
//
// We mean it.
//

#ifndef OSDAB_ZIP_P__H
#define OSDAB_ZIP_P__H

#include "zip.h"
#include "zipentry_p.h"

#include <QtGlobal>
#include <QFileInfo>

/*!
  zLib authors suggest using larger buffers (128K or 256K) for (de)compression (especially for inflate())
  we use a 256K buffer here - if you want to use this code on a pre-iceage mainframe please change it ;)
*/
#define ZIP_READ_BUFFER (256*1024)

class ZipPrivate
{
public:
  ZipPrivate();
  virtual ~ZipPrivate();

  QMap<QString,ZipEntryP*>* headers;

  QIODevice* device;

  char buffer1[ZIP_READ_BUFFER];
  char buffer2[ZIP_READ_BUFFER];

  unsigned char* uBuffer;

  const quint32* crcTable;

  QString comment;
  QString password;

  Zip::ErrorCode createArchive(QIODevice* device);
  Zip::ErrorCode closeArchive();
  void reset();

  bool zLibInit();

  Zip::ErrorCode createEntry(const QFileInfo& file, const QString& root, Zip::CompressionLevel level);
  Zip::CompressionLevel detectCompressionByMime(const QString& ext);

  inline void encryptBytes(quint32* keys, char* buffer, qint64 read);

  inline void setULong(quint32 v, char* buffer, unsigned int offset);
  inline void updateKeys(quint32* keys, int c) const;
  inline void initKeys(quint32* keys) const;
  inline int decryptByte(quint32 key2) const;

  inline QString extractRoot(const QString& p);
};

#endif // OSDAB_ZIP_P__H
```

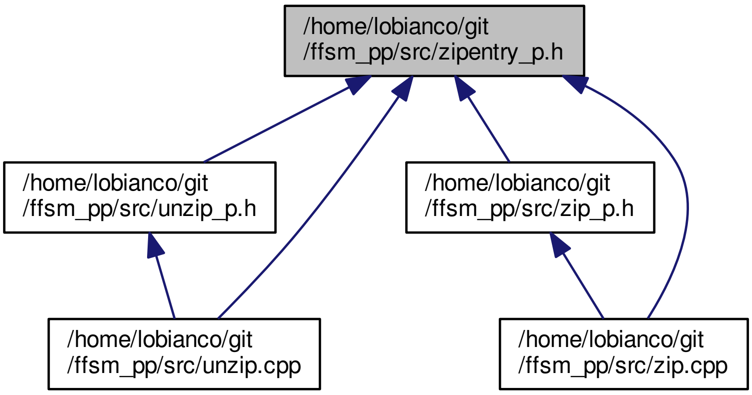
5.155 /home/lobianco/git/ffsm_pp/src/zipentry_p.h File Reference

```
#include <QtGlobal>
#include <QString>
```

Include dependency graph for zipentry_p.h:



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



Classes

- class ZipEntryP

5.156 zipentry_p.h

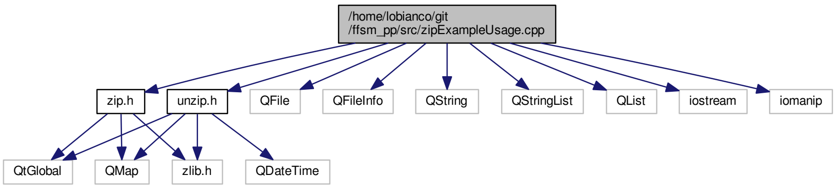
```
/****************************************************************************
** Filename: ZipEntryP.h
** Last updated [dd/mm/yyyy]: 28/01/2007
**
** Wrapper for a ZIP local header.
**
** Some of the code has been inspired by other open source projects,
** (mainly Info-Zip and Gilles Vollant's minizip).
** Compression and decompression actually uses the zlib library.
**
** Copyright (C) 2007 Angius Fabrizio. All rights reserved.
**
** This file is part of the OSDaB project (http://osdab.sourceforge.net/).
```

```
**
** This file may be distributed and/or modified under the terms of the
** GNU General Public License version 2 as published by the Free Software
** Foundation and appearing in the file LICENSE.GPL included in the
** packaging of this file.
**
** This file is provided AS IS with NO WARRANTY OF ANY KIND, INCLUDING THE
** WARRANTY OF DESIGN, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
**
** See the file LICENSE.GPL that came with this software distribution or
** visit http://www.gnu.org/copyleft/gpl.html for GPL licensing information.
**
**********************************************************************/

//
//  W A R N I N G
//  -------------
//
// This file is not part of the Zip/UnZip API.  It exists purely as an
// implementation detail. This header file may change from version to
// version without notice, or even be removed.
//
// We mean it.
//

#ifndef OSDAB_ZIPENTRY_P__H
#define OSDAB_ZIPENTRY_P__H

#include <QtGlobal>
#include <QString>

class ZipEntryP
{
public:
  ZipEntryP()
  {
    lhOffset = 0;
    dataOffset = 0;
    gpFlag[0] = gpFlag[1] = 0;
    compMethod = 0;
    modTime[0] = modTime[1] = 0;
    modDate[0] = modDate[1] = 0;
    crc = 0;
    szComp = szUncomp = 0;
    lhEntryChecked = false;
  }

  quint32 lhOffset;      // Offset of the local header record for this entry
  quint32 dataOffset;      // Offset of the file data for this entry
  unsigned char gpFlag[2];  // General purpose flag
  quint16 compMethod;       // Compression method
  unsigned char modTime[2];  // Last modified time
  unsigned char modDate[2];  // Last modified date
  quint32 crc;        // CRC32
  quint32 szComp;        // Compressed file size
  quint32 szUncomp;      // Uncompressed file size
  QString comment;      // File comment

  bool lhEntryChecked;    // Is true if the local header record for this entry has been
 parsed

  inline bool isEncrypted() const { return gpFlag[0] & 0x01; }
  inline bool hasDataDescriptor() const { return gpFlag[0] & 0x08; }
};

#endif // OSDAB_ZIPENTRY_P__H
```

5.157 /home/lobianco/git/ffsm_pp/src/zipExampleUsage.cpp File Reference

```
#include "zip.h"
#include "unzip.h"
#include <QFile>
#include <QFileInfo>
#include <QString>
#include <QStringList>
#include <QList>
#include <iostream>
#include <iomanip>
```

Include dependency graph for zipExampleUsage.cpp:



Functions

- void invalidCMD ()
- bool decompress (const QString &file, const QString &out, const QString &pwd)
- bool compress (const QString &zip, const QString &dir, const QString &pwd)
- bool listFiles (const QString &file, const QString &pwd)
- int main (int argc, char ∗∗argv)

5.157.1 Function Documentation

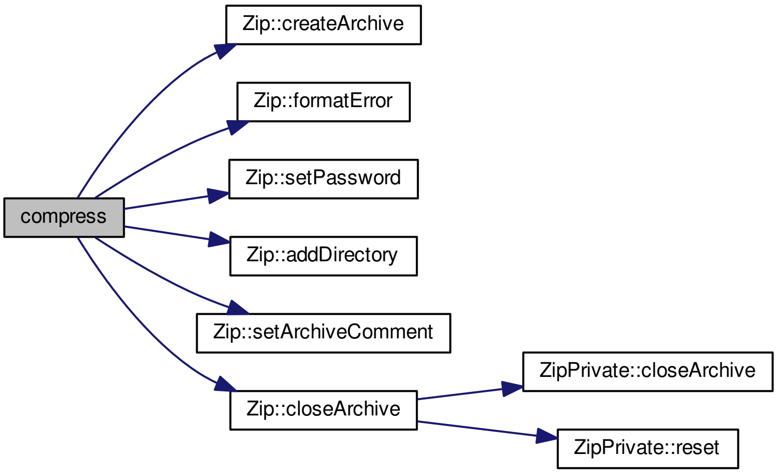
5.157.1.1 bool compress (const QString & *zip,* const QString & *dir,* const QString & *pwd*)

Definition at line 182 of file zipExampleUsage.cpp.

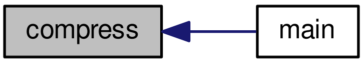
Referenced by main().

```
{
  QFileInfo fi(dir);
  if (!fi.isDir())
  {
    cout << "Directory does not exist." << endl << endl;
    return false;
  }

  Zip::ErrorCode ec;
  Zip uz;

  ec = uz.createArchive(zip);
  if (ec != Zip::Ok)
  {
    cout << "Unable to create archive: " << uz.formatError(ec).toAscii().data() << endl << endl;
    return false;
  }

  uz.setPassword(pwd);
  ec = uz.addDirectory(dir);
  if (ec != Zip::Ok)
  {
    cout << "Unable to add directory: " << uz.formatError(ec).toAscii().data() << endl << endl;
  }

  uz.setArchiveComment("This archive has been created using OSDaB Zip
 (http://osdab.sourceforge.net/).");

  if (uz.closeArchive() != Zip::Ok)
  {
    cout << "Unable to close the archive: " << uz.formatError(ec).toAscii().data() << endl <<
endl;
  }

  return ec == Zip::Ok;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



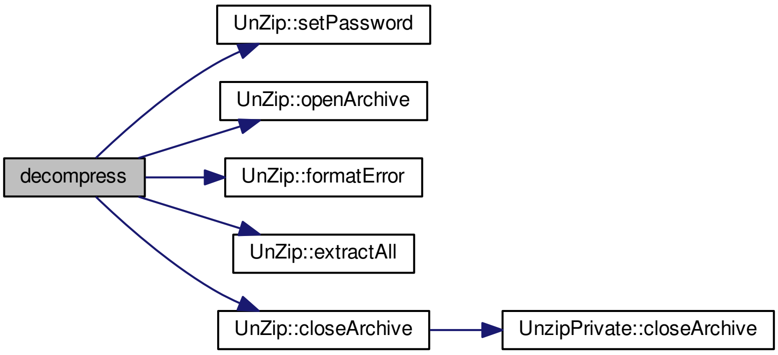
5.157.1.2 bool decompress (const QString & *file*, const QString & *out*, const QString & *pwd*)

Definition at line 149 of file zipExampleUsage.cpp.

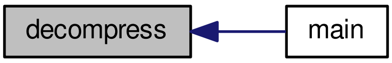
Referenced by main().

```
{

  if (!QFile::exists(file))
  {
    cout << "File does not exist." << endl << endl;
    return false;
  }

  UnZip::ErrorCode ec;
  UnZip uz;

  if (!pwd.isEmpty())
    uz.setPassword(pwd);

  ec = uz.openArchive(file);
  if (ec != UnZip::Ok)
  {
    cout << "Failed to open archive: " << uz.formatError(ec).toAscii().data() << endl << endl;
    return false;
  }

  ec = uz.extractAll(out);
  if (ec != UnZip::Ok)
  {
    cout << "Extraction failed: " << uz.formatError(ec).toAscii().data() << endl << endl;
    uz.closeArchive();
    return false;
  }

  return true;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:



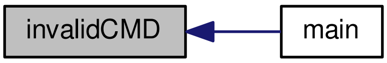
5.157.1.3 void invalidCMD ( )

Definition at line 140 of file zipExampleUsage.cpp.

Referenced by main().

```
{
    cout << "Invalid command line. Usage:" << endl;
    cout << "Compression: zip [-p PWD] DIRECTORY" << endl;
    cout << "List files: zip -l [-p PWD] ZIPFILE" << endl;
    cout << "Decompression: zip -d [-p PWD] ZIPFILE OUTPUT_DIR" << endl << endl;
    exit(-1);
}
```

Here is the caller graph for this function:



5.157.1.4 bool listFiles (const QString & *file*, const QString & *pwd*)

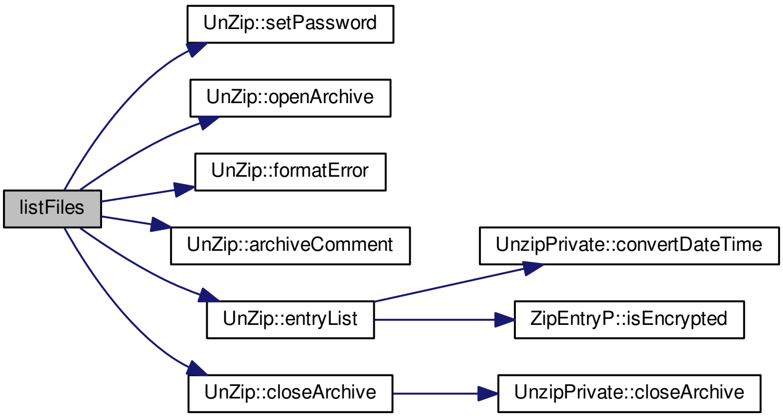
Definition at line 218 of file zipExampleUsage.cpp.

Referenced by main().

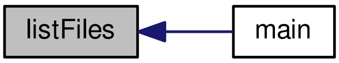
```
{
  if (!QFile::exists(file))
  {
    cout << "File does not exist." << endl << endl;
    return false;
  }

  UnZip::ErrorCode ec;
  UnZip uz;

  if (!pwd.isEmpty())
    uz.setPassword(pwd);

  ec = uz.openArchive(file);
  if (ec != UnZip::Ok)
  {
    cout << "Unable to open archive: " << uz.formatError(ec).toAscii().data() << endl << endl;
    return false;
  }

  QString comment = uz.archiveComment();
  if (!comment.isEmpty())
    cout << "Archive comment: " << comment.toAscii().data() << endl << endl;

  QList<UnZip::ZipEntry> list = uz.entryList();
  if (list.isEmpty())
  {
    cout << "Empty archive.";
  }
  else
  {
    cout.setf(ios::left);
    cout << setw(40) << "Filename";
    cout.unsetf(ios::left);
    cout << setw(10) << "Size" << setw(10) << "Ratio" << setw(10) << "CRC32" << endl;
    cout.setf(ios::left);
    cout << setw(40) << "--------";
    cout.unsetf(ios::left);
    cout << setw(10) << "----" << setw(10) << "-----" << setw(10) << "-----" << endl;

    for (int i = 0; i < list.size(); ++i)
    {
      const UnZip::ZipEntry& entry = list.at(i);

      double ratio = entry.uncompressedSize == 0 ? 0 : 100 - (double) entry.compressedSize * 100 / (double) entry.uncompressedSize;

      QString ratioS = QString::number(ratio, 'f', 2).append("%");
      QString crc;
      crc = crc.sprintf("%X", entry.crc32).rightJustified(8, '0');
      QString file = entry.filename;
      int idx = file.lastIndexOf("/");
      if (idx >= 0 && idx != file.length()-1)
        file = file.right(file.length() - idx - 1);
      file = file.leftJustified(40, ' ', true);

      if (entry.encrypted)
```

```
        file.append("*");

      cout << setw(40) << file.toAscii().data() << setw(10) << entry.uncompressedSize << setw(10) << ratioS.toAscii().data() << setw(10) << crc.toAscii().data() << endl;
    }
  }

  uz.closeArchive();
  return true;
}
```

Here is the call graph for this function:



Here is the caller graph for this function:

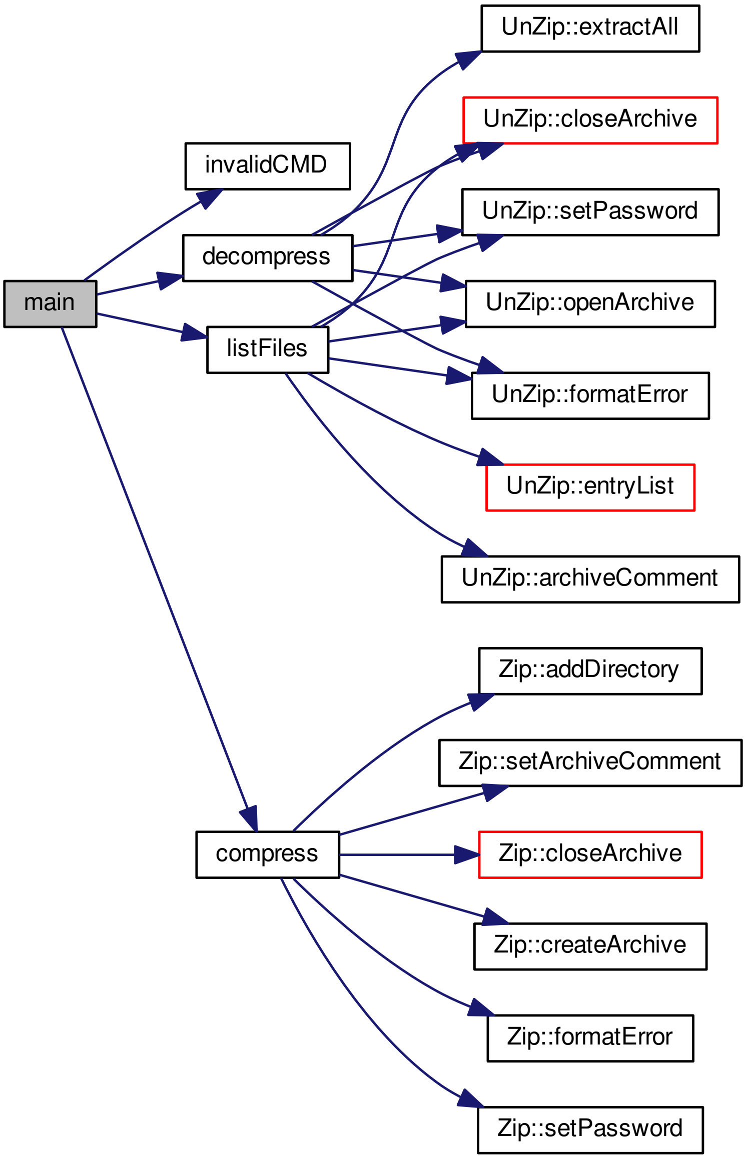


5.157.1.5 int main (int *argc*, char ** *argv*)

Definition at line 44 of file zipExampleUsage.cpp.

```
{
  if (argc < 3)
  {
    cout << "Test routine for the OSDaB Project Zip/UnZip classes" << endl << endl;
    cout << "Compression: zip [-p PWD] ZIPFILE DIRECTORY" << endl;
    cout << "List files: zip -l [-p PWD] ZIPFILE" << endl;
    cout << "Decompression: zip -d [-p PWD] ZIPFILE OUTPUT_DIR" << endl << endl;
    cout << "(C) 2007 Angius Fabrizio\nLicensed under the terms of the GNU GPL Version 2 or later" << endl;
    return -1;
  }

  QString fname;
  QString dname;
  QString pwd;

  bool resOK = true;

  if (strlen(argv[1]) == 2 && argv[1][0] == '-')
  {
    switch (argv[1][1])
    {
      case 'd':
      {
        if (argc >= 6)
        {
          if (strcmp(argv[2], "-p") == 0)
          {
            pwd = QString(argv[3]);
            fname = QString(argv[4]);
            dname = QString(argv[5]);
          }
          else invalidCMD();
        }
        else if (argc >= 4)
        {
          fname = QString(argv[2]);
          dname = QString(argv[3]);
        }
        else invalidCMD();

        resOK = decompress(fname, dname, pwd);
      }
      break;
      case 'l':
      {
        if (argc >= 5)
        {
          if (strcmp(argv[2], "-p") == 0)
          {
            pwd = QString(argv[3]);
            fname = QString(argv[4]);
          }
          else invalidCMD();
        }
        else if (argc >= 3)
        {
          fname = QString(argv[2]);
        }
        else invalidCMD();

        resOK = listFiles(fname, pwd);
      }
      break;
      case 'p':
      {
        if (argc >= 5)
        {
          pwd = QString(argv[2]);
          fname = QString(argv[3]);
          dname = QString(argv[4]);
        }
        else invalidCMD();

        resOK = compress(fname, dname, pwd);
      }
      break;
      default: invalidCMD();
    }
  }
  else
  {
    // no parameters -- compress directly
    resOK = compress(QString(argv[1]), QString(argv[2]), 0);
  }


  if (!resOK)
  {
    cout << "Sorry, some error occurred!" << endl;
    return -1;
  }

  return 0;
}
```

Here is the call graph for this function:



5.158 zipExampleUsage.cpp

```
/****************************************************************************
** Filename: main.cpp
** Last updated [dd/mm/yyyy]: 01/02/2007
**
** Test routine for the Zip and UnZip classed.
**
** Copyright (C) 2007 Angius Fabrizio. All rights reserved.
**
** This file is part of the OSDaB project (http://osdab.sourceforge.net/).
**
** This file may be distributed and/or modified under the terms of the
** GNU General Public License version 2 as published by the Free Software
** Foundation and appearing in the file LICENSE.GPL included in the
** packaging of this file.
**
** This file is provided AS IS with NO WARRANTY OF ANY KIND, INCLUDING THE
** WARRANTY OF DESIGN, MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
**
** See the file LICENSE.GPL that came with this software distribution or
** visit http://www.gnu.org/copyleft/gpl.html for GPL licensing information.
**
**********************************************************************/

#include "zip.h"
#include "unzip.h"

#include <QFile>
#include <QFileInfo>

#include <QString>
#include <QStringList>
#include <QList>

#include <iostream>
#include <iomanip>

void invalidCMD();
bool decompress(const QString& file, const QString& out, const QString& pwd);
bool compress(const QString& zip, const QString& dir, const QString& pwd);
bool listFiles(const QString& file, const QString& pwd);

using namespace std;

int main(int argc, char** argv)
{
  if (argc < 3)
  {
    cout << "Test routine for the OSDaB Project Zip/UnZip classes" << endl << endl;
    cout << "Compression: zip [-p PWD] ZIPFILE DIRECTORY" << endl;
    cout << "List files: zip -l [-p PWD] ZIPFILE" << endl;
    cout << "Decompression: zip -d [-p PWD] ZIPFILE OUTPUT_DIR" << endl << endl;
    cout << "(C) 2007 Angius Fabrizio\nLicensed under the terms of the GNU GPL Version 2 or later" << endl;
    return -1;
  }

  QString fname;
  QString dname;
  QString pwd;

  bool resOK = true;

  if (strlen(argv[1]) == 2 && argv[1][0] == '-')
  {
    switch (argv[1][1])
    {
      case 'd':
      {
        if (argc >= 6)
        {
          if (strcmp(argv[2], "-p") == 0)
          {
            pwd = QString(argv[3]);
            fname = QString(argv[4]);
            dname = QString(argv[5]);
          }
          else invalidCMD();
        }
        else if (argc >= 4)
        {
          fname = QString(argv[2]);
          dname = QString(argv[3]);
        }
        else invalidCMD();

        resOK = decompress(fname, dname, pwd);
      }
      break;
      case 'l':
      {
        if (argc >= 5)
        {
          if (strcmp(argv[2], "-p") == 0)
          {
            pwd = QString(argv[3]);
            fname = QString(argv[4]);
          }
          else invalidCMD();
        }
        else if (argc >= 3)
    {
     fname = QString(argv[2]);
    }
    else invalidCMD();

    resOK = listFiles(fname, pwd);
   }
   break;
   case 'p':
   {
    if (argc >= 5)
    {
     pwd = QString(argv[2]);
     fname = QString(argv[3]);
     dname = QString(argv[4]);
    }
    else invalidCMD();

    resOK = compress(fname, dname, pwd);
   }
   break;
   default: invalidCMD();
  }
 }
 else
 {
  // no parameters -- compress directly
  resOK = compress(QString(argv[1]), QString(argv[2]), 0);
 }


 if (!resOK)
 {
  cout << "Sorry, some error occurred!" << endl;
  return -1;
 }

 return 0;
}

void invalidCMD()
{
  cout << "Invalid command line. Usage:" << endl;
  cout << "Compression: zip [-p PWD] DIRECTORY" << endl;
  cout << "List files: zip -l [-p PWD] ZIPFILE" << endl;
  cout << "Decompression: zip -d [-p PWD] ZIPFILE OUTPUT_DIR" << endl << endl;
  exit(-1);
}

bool decompress(const QString& file, const QString& out, const QString& pwd)
{

 if (!QFile::exists(file))
 {
  cout << "File does not exist." << endl << endl;
  return false;
 }

 UnZip::ErrorCode ec;
 UnZip uz;

 if (!pwd.isEmpty())
  uz.setPassword(pwd);

 ec = uz.openArchive(file);
 if (ec != UnZip::Ok)
 {
  cout << "Failed to open archive: " << uz.formatError(ec).toAscii().data() << endl << endl;
  return false;
 }

 ec = uz.extractAll(out);
 if (ec != UnZip::Ok)
 {
  cout << "Extraction failed: " << uz.formatError(ec).toAscii().data() << endl << endl;
  uz.closeArchive();
  return false;
 }

 return true;
}

bool compress(const QString& zip, const QString& dir, const QString& pwd)
{
 QFileInfo fi(dir);
 if (!fi.isDir())
 {
  cout << "Directory does not exist." << endl << endl;
  return false;
 }

 Zip::ErrorCode ec;
 Zip uz;

 ec = uz.createArchive(zip);
 if (ec != Zip::Ok)
 {
  cout << "Unable to create archive: " << uz.formatError(ec).toAscii().data() << endl << endl;
  return false;
 }

 uz.setPassword(pwd);
 ec = uz.addDirectory(dir);
 if (ec != Zip::Ok)
 {
  cout << "Unable to add directory: " << uz.formatError(ec).toAscii().data() << endl << endl;
 }

 uz.setArchiveComment("This archive has been created using OSDaB Zip (http://osdab.sourceforge.net/).");

 if (uz.closeArchive() != Zip::Ok)
 {
  cout << "Unable to close the archive: " << uz.formatError(ec).toAscii().data() << endl << endl;
 }

 return ec == Zip::Ok;
}

bool listFiles(const QString& file, const QString& pwd)
{
 if (!QFile::exists(file))
 {
  cout << "File does not exist." << endl << endl;
  return false;
 }

 UnZip::ErrorCode ec;
 UnZip uz;

 if (!pwd.isEmpty())
  uz.setPassword(pwd);

 ec = uz.openArchive(file);
 if (ec != UnZip::Ok)
 {
  cout << "Unable to open archive: " << uz.formatError(ec).toAscii().data() << endl << endl;
  return false;
 }

 QString comment = uz.archiveComment();
 if (!comment.isEmpty())
  cout << "Archive comment: " << comment.toAscii().data() << endl << endl;

 QList<UnZip::ZipEntry> list = uz.entryList();
 if (list.isEmpty())
 {
  cout << "Empty archive.";
 }
 else
 {
  cout.setf(ios::left);
  cout << setw(40) << "Filename";
  cout.unsetf(ios::left);
  cout << setw(10) << "Size" << setw(10) << "Ratio" << setw(10) << "CRC32" << endl;
  cout.setf(ios::left);
  cout << setw(40) << "--------";
  cout.unsetf(ios::left);
  cout << setw(10) << "----" << setw(10) << "-----" << setw(10) << "-----" << endl;

  for (int i = 0; i < list.size(); ++i)
  {
    const UnZip::ZipEntry& entry = list.at(i);

    double ratio = entry.uncompressedSize == 0 ? 0 : 100 - (double) entry.compressedSize * 100 / (double) entry.uncompressedSize;

    QString ratioS = QString::number(ratio, ’f’, 2).append("%");
    QString crc;
    crc = crc.sprintf("%X", entry.crc32).rightJustified(8, ’0’);
    QString file = entry.filename;
    int idx = file.lastIndexOf("/");
    if (idx >= 0 && idx != file.length()-1)
```

```
        file = file.right(file.length() - idx - 1);
      file = file.leftJustified(40, ' ', true);

      if (entry.encrypted)
        file.append("*");

      cout << setw(40) << file.toAscii().data() << setw(10) << entry.uncompressedSize << setw(10) << ratioS.toAscii().data() << setw(10) << crc.toAscii().data() << endl;
    }
  }

  uz.closeArchive();
  return true;
}
```

5.159 /home/lobianco/git/ffsm_pp/TODO File Reference

5.160 /home/lobianco/git/ffsm_pp/TODO

```
[2007.12.28 Antonello]
TODO list is now directly embedded within the source code. Use an IDE that can search and list them automatically (like KDevelop)



```

5.161 /home/lobianco/git/ffsm_pp/windowsInstallerScript.nsi File Reference

5.162 /home/lobianco/git/ffsm_pp/windowsInstallerScript.nsi

```
; FFSM++ Windows Instal Script
; Writen to be compiled with NSIS
; Antonello Lobianco, 2015

;--------------------------------
; General
 ; Things that will likely need to be changed...
 !define VERSIONSTRING "1.1.0"               ; Pythiaversion number (as string). Used to put each version on a separate folder.

 OutFile "ffsm_pp_${VERSIONSTRING}_setup.exe" ; Filename of the outputted installer
 !define MINGWDIR "C:\MinGW\bin" ; Directory where thre MinGW DLL is located
 !define QTDIR "C:\Qt\4.8.2\bin"   ; Directory where the Qt run-time DLLs are located
 !define EXEDIR "."                ; Directory where the EXE file is located

 !include "MUI.nsh" ; Include Modern UI
 Name "FFSM++" ; Name (?)
 InstallDir "$PROGRAMFILES\FFSM\${VERSIONSTRING}" ; Default installation directory
 !define APP_NAME "FFSM++" ; Application name (mainly for links)
 !define APP_FNAME "FFSM++ ${VERSIONSTRING}" ; Application name with version(mainly for links)
 !define MAIN_APP_EXE "ffsm.exe" ; Filename of executable
 !define WEB_SITE "http://ffsm-project.org" ; We-site address
 Var OPTIONALDATA ; We'll use this variable to put links on sample data only if users have selected to intall sample data
 Var LOCALDOC ;If user has selected to install documentation
 Var SM_Folder ; Application shortcuts main folder
 SetCompressor ZLIB ; Compression used
 XPStyle on ; Style of the Wizard (look it doesn't change anything)

;--------------------------------
; Interface Settings ( we define all settings before building the pages - next points)

 !define MUI_ABORTWARNING ; We give a confirmation warning before let the user abort the installation proces
 !define MUI_ICON "src\imgs\ffsm.ico" ; icon for the installer
 !define MUI_UNICON "src\imgs\ffsm.ico" ; icon for the uninstaller
 !define MUI_WELCOMEFINISHPAGE_BITMAP "src\imgs\beech.bmp" ; image for the welcome and finish page
 !define MUI_UNWELCOMEFINISHPAGE_BITMAP "src\imgs\beech.bmp" ; image for the welcome and finish page (uninstaller)
 !define MUI_COMPONENTSPAGE_SMALLDESC ; section description on the bottom instead on the default right
 !define REG_START_MENU "Start Menu Folder" ; ??
 !define MUI_STARTMENUPAGE_DEFAULTFOLDER "FFSM\${VERSIONSTRING}" ; Default folder where to prompt the user to place links
 !define MUI_FINISHPAGE_RUN "$INSTDIR\${MAIN_APP_EXE}" ; What propose the user to do after the installation is completed
 !define MUI_WELCOMEPAGE_TEXT "Version ${VERSIONSTRING}\n\n FFSM++ is a flexible, spatially explicit, coupled resource and economic simulator of the Forest Sector, designed for long-term simulations of effects of government policies over different forest systems.\n\n This Wizard will guide you the installation of FFSM++. \n\n Press Next to start the installation."

;--------------------------------
; Installer pages (steps)

 !insertmacro MUI_PAGE_WELCOME ; Welcome page
 !insertmacro MUI_PAGE_LICENSE "COPYING" ; Accept licence page
 !insertmacro MUI_PAGE_COMPONENTS ; Choose installation components (pieces)
 !insertmacro MUI_PAGE_DIRECTORY ; Directory where to install
 !insertmacro MUI_PAGE_STARTMENU Application $SM_Folder ; Write links (and where)
 !insertmacro MUI_PAGE_INSTFILES ; Installing the files
 !insertmacro MUI_PAGE_FINISH ; "Done!" page
 !insertmacro MUI_UNPAGE_CONFIRM ; Confirmation request before uninstalling
 !insertmacro MUI_UNPAGE_INSTFILES ; Uninstalling files

;--------------------------------
; Languages

 !insertmacro MUI_LANGUAGE "English" ; ??

;--------------------------------
; Installer Sections (components that user can choose if install it or not)

Section "Main program" MainProgram
 SectionIn RO  ; Read only - the user can not delesect it !
 SetOutPath "$INSTDIR" ; Where files need to be installed
 File "${EXEDIR}\ffsm.exe" ; Adding this file or directory to the list of files to be installed
 File "${MINGWDIR}\mingwm10.dll"
 File "${MINGWDIR}\pthreadGC2.dll"
 File "${MINGWDIR}\libgcc_s_dw2-1.dll"
 File "${MINGWDIR}\libgfortran-3.dll"
 File "${MINGWDIR}\libquadmath-0.dll"
 File "${MINGWDIR}\libstdc++-6.dll"
 File "src\ThirdParty\win32\lib\libadolc-1.dll"
 File "AUTHORS"
 File "COPYING"
 File "NEWS"
 File "README"
 WriteUninstaller "$INSTDIR\Uninstall_ffsm.exe" ; Creating the uninstaller
SectionEnd

Section "Run-time libraries" Qt
 SetOutPath "$INSTDIR"
 File "${QTDIR}\QtCore4.dll"
 File "${QTDIR}\QtGui4.dll"
 File "${QTDIR}\QtXml4.dll"
SectionEnd

Section "Sample data" Data
 SetOutPath "$INSTDIR\data"
 File /r "data\*" ; Adding this file or directory to the list of files to be installed. /r means "recursively"
 StrCpy $OPTIONALDATA "true" ; Saving the fact the user has chosen to install the sample data so later on me make the links
SectionEnd

Section /o "Source" Src   ; option /o means optional - unselected by default
 SetOutPath "$INSTDIR\src"
 File "src\*.h"
 File "src\*.cpp"
 File "src\*.pro"
SectionEnd

Section /o "Documentation" Doc   ;
 ;not yet ready...
 SetOutPath "$INSTDIR\doc"
 File "doc\Install run and develop instructions.pdf"
 File "doc\Input and output data management.pdf"
 File "doc\Reference manual.pdf"
 StrCpy $LOCALDOC "true"
SectionEnd

;--------------------------------
; Component Descriptions

 ; Creating "Language strings" objects for each section...
 LangString DESC_MainProgram ${LANG_ENGLISH} "Main FFSM++ files"
 LangString DESC_Qt ${LANG_ENGLISH} "Run-time graphical libraries. Unselect this section only if you already have Qt 4.X installed on your PC"
 LangString DESC_Data ${LANG_ENGLISH} "Sample input data (recommended)"
 LangString DESC_Src ${LANG_ENGLISH} "FFSM++ C++ source code (not needed to run the program)"
 LangString DESC_Doc ${LANG_ENGLISH} "Local copy of the documentation (for more doc refer to the
 site)"

 ;Assign "Language strings" objects to sections
 !insertmacro MUI_FUNCTION_DESCRIPTION_BEGIN
   !insertmacro MUI_DESCRIPTION_TEXT ${MainProgram} $(DESC_MainProgram)
   !insertmacro MUI_DESCRIPTION_TEXT ${Qt} $(DESC_Qt)
   !insertmacro MUI_DESCRIPTION_TEXT ${Data} $(DESC_Data)
   !insertmacro MUI_DESCRIPTION_TEXT ${Src} $(DESC_Src)
   !insertmacro MUI_DESCRIPTION_TEXT ${Doc} $(DESC_Doc)
 !insertmacro MUI_FUNCTION_DESCRIPTION_END

;--------------------------------
; Links & icons

Section -Icons_Reg
 SetOutPath "$INSTDIR" ; Where files need to be installed by default

 !ifdef REG_START_MENU ; If the user has chosen to make links
   !insertmacro MUI_STARTMENU_WRITE_BEGIN Application
     ; Create directory for the shrtcuts ($M_Folder has been chosen by the user)
     CreateDirectory "$SMPROGRAMS\$SM_Folder"
     ; Shortcut to the main program
     CreateShortCut "$SMPROGRAMS\$SM_Folder\${APP_NAME}.lnk" "$INSTDIR\${MAIN_APP_EXE}"
     ; Desktop shortcut to the main program
     CreateShortCut "$DESKTOP\${APP_FNAME}.lnk" "$INSTDIR\${MAIN_APP_EXE}"
     ; Shortcut to the uninstaller
     CreateShortCut "$SMPROGRAMS\$SM_Folder\Uninstall ${APP_NAME}.lnk" "$INSTDIR\Uninstall_ffsm.exe"
     ${If} $OPTIONALDATA == 'true' ; If user has installed the sample data
       ; Shortcut to data file
       CreateShortCut "$SMPROGRAMS\$SM_Folder\Edit sample data.lnk" "$INSTDIR\data\ffsmInput.ods"
     ${EndIf}
     ${If} $LOCALDOC == 'true' ; If user has installed a local copy of the documentation
       ; Shortcut to user manual file
       CreateShortCut "$SMPROGRAMS\$SM_Folder\Install run and develop instructions.lnk"
 "$INSTDIR\doc\Install run and develop instructions.pdf"
       ; Shortcut to data management
       CreateShortCut "$SMPROGRAMS\$SM_Folder\Input and output data management.lnk"
 "$INSTDIR\doc\Input and output data management.pdf"
       ; Shortcut to the reference manual
       CreateShortCut "$SMPROGRAMS\$SM_Folder\Reference Manual.lnk" "$INSTDIR\doc\Reference
 manual.pdf"
     ${EndIf}
     ; Write internet shortcut to the main web-site and link it from the other START MENU shortcuts..
     WriteIniStr "$INSTDIR\${APP_NAME} website.url" "InternetShortcut" "URL" "${WEB_SITE}"
     CreateShortCut "$SMPROGRAMS\$SM_Folder\${APP_NAME} Website.lnk" "$INSTDIR\${APP_NAME}
 website.url"
     ; Write internet shortcut to the on-line documentation and link it from the other START MENU
 shortcuts..
     WriteIniStr "$INSTDIR\${APP_NAME} documentation.url" "InternetShortcut" "URL" "${WEB_SITE}"
     CreateShortCut "$SMPROGRAMS\$SM_Folder\${APP_NAME} Documentation.lnk" "$INSTDIR\${APP_NAME}
 documentation.url"
   !insertmacro MUI_STARTMENU_WRITE_END
 !endif
SectionEnd

;--------------------------------
;Uninstaller Section

Section "Uninstall"
 RMDir /r "$INSTDIR" ; Recursivelly remove all files in the install directory
 RMDir /r "$SMPROGRAMS\$SM_Folder" ; Remove links in the Start Menu ..doesn't works !!!
 Delete "$DESKTOP\${APP_NAME}.lnk" ; Remove desktop link
 ;DeleteRegKey /ifempty HKCU "Software\Pythia"
SectionEnd

```