MOHIDLagrangian

0.01

Generated by Doxygen 1.8.14

Contents

Chapter 1

MOHIDLagrangian

MOHIDLagragian is a both a library for the MOHID Water Modelling System and a standalone program. The library implements all the necessary tools to generate a comprehensive Lagrangian tracer model, with sources, sinks, particle types and several options for forcing and I/O.

The MOHIDLagrangian program is a specific implementation of the library, designed as a post-processing or online tool, ready to be forced with other models.

Help, Bugs, Feedback

If you need help with MOHIDLagrangian or MOHID, want to keep up with progress, chat with developers or ask any other questions about MOHID, you can hang out by mail: general@mohid.com or consult our MOHID wiki. You can also subscribe to our MOHID forum. To report bugs, please create a GitHub issue or contact any developers. More information consult http://www.mohid.com

License

GNU General Public License. See the GNU General Public License web page for more information.

Chapter 2

Modules Index

2.1 Modules List

Here is a list of all modules with brief descriptions:

Chapter 3

Data Type Index

3.1 Class Hierarchy

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

Chapter 4

Data Type Index

4.1 Data Types List

Here are the data types with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Module Documentation

6.1 about_mod Module Reference

Module to print version, licence, preambles.

Functions/Subroutines

- subroutine, public printlicpreamble

  Public licence and preamble printer routine.

Variables

- type(string) version
- type(string) author
- type(string) date

6.1.1 Detailed Description

Module to print version, licence, preambles.

Author

Ricardo Birjukovs Canelas

6.1.2 Function/Subroutine Documentation

6.1.2.1 printlicpreamble()

```
subroutine, public about_mod::printlicpreamble ( )
```

Public licence and preamble printer routine.

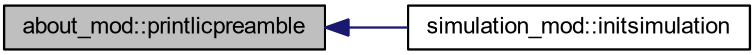
Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 44 of file about.f90.

```
 implicit none
 type(string) :: outext

 version  ="v0.0.4"
 author   ="R. Birjukovs Canelas"
 date     ="29-05-2018"

 outext = ' __  __  ___  _   _ ___ ____  _                                    _            '//
new_line('a')//&
          ' |  \/  |/ _ \| | | |_ _|  _ \| |     __ _  __ _ _ __ __ _ _ __   __ _(_) __ _ _ __  '//
new_line('a')//&
          ' | |\/| | | | | |_| || || | | | |    / _` |/ _` | __/  _` |  _ \ / _` | |/ _` |  _ \ '//
new_line('a')//&
          ' | |  | | |_| |  _  || || |_| | |___| (_| | (_| | | | (_| | | | | (_| | | (_| | | | |'//
new_line('a')//&
          ' |_|  |_|\___/|_| |_|___|____/|_____\__,_|\__, |_|  \__,_|_| |_|\__, |_|\__,_|_| |_|'//
new_line('a')//&
          '                                          |___/                  |___/              '//
new_line('a')//&

     '  <MOHIDLagrangian> Copyright (C) 2018 by'//new_line('a')//&
     '  R. Birjukovs Canelas, R. Neves, F. Campuzano, H. de Pablo Lenonardo'//new_line('a')//&
     ''//new_line('a')//&
     '  MARETEC - Research Centre for Marine, Environment and Technology'//new_line('a')//&
     ''//new_line('a')//&
     '  MOHIDLagrangian is free software: you can redistribute it and/or'//new_line('a')//&
     '  modify it under the terms of the GNU General Public License as'//new_line('a')//&
     '  published by the Free Software Foundation, either version 3 of'//new_line('a')//&
     '  the License, or (at your option) any later version.'//new_line('a')//&
     ''//new_line('a')//&
     '  MOHIDLagrangian is distributed WITHOUT ANY WARRANTY; without even'//new_line('a')//&
     '  the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR'//new_line('a')//&
     '  PURPOSE. See the GNU General Public License for more details.'//new_line('a')//&
     ''//new_line('a')//&
     '  You should have received a copy of the GNU General Public License,'//new_line('a')//&
     '  along with MOHIDLagrangian. If not, see <http://www.gnu.org/licenses/>.,'//new_line('a')//&
     ''//new_line('a')//&
     ''//new_line('a')//&
     'MOHIDLagrangian '//version//' ('//author//') ('//date//')'//new_line('a')//&
     '=====================================================================' 

 !call Log%put(outext,.false.)
 call log%put(outext,.false.)

```

Here is the caller graph for this function:



6.1.3 Variable Documentation

6.1.3.1 author

```
type(string) about_mod::author [private]
```

Definition at line 31 of file about.f90.

```
31     type(string) :: author
```

6.1.3.2 date

```
type(string) about_mod::date [private]
```

Definition at line 32 of file about.f90.

```
32     type(string) :: date
```

6.1.3.3 version

```
type(string) about_mod::version [private]
```

Definition at line 30 of file about.f90.

```
30     type(string) :: version
```

6.2 abstract_container_array_mod Module Reference

Module that defines an unlimited polymorphic container class and related methods. A container is a fundamental entity allowing to build data structures such as lists and arrays. This is an abstract type, so a derived type must be defined for any specific contents that may be required. Those derived types should provide type-specific methods that require type-guards, such as printing.

Data Types

- type container_array

Functions/Subroutines

- class(∗) function, pointer getvalue (this, index)

  Method that returns returns the requested entry (pointer)
- subroutine putvalue (this, index, value)

  Method that stores a value on the requested index.
- integer function getlength (this)

  Method that returns the length of the array.
- subroutine resizearray (this, newsize)

  Method that grows (adds empty space) or shrinks (discards the last entries) of the array. Use sparsely as this might get expensive for large array operations. Should think of a way to use move_alloc()
- subroutine initarray (this, entries, tocopy)

  Method that allocates the container array. Deallocates if already allocated.

6.2.1 Detailed Description

Module that defines an unlimited polymorphic container class and related methods. A container is a fundamental entity allowing to build data structures such as lists and arrays. This is an abstract type, so a derived type must be defined for any specific contents that may be required. Those derived types should provide type-specific methods that require type-guards, such as printing.

Author

Ricardo Birjukovs Canelas

6.2.2 Function/Subroutine Documentation

6.2.2.1 getlength()

```
integer function abstract_container_array_mod::getlength (
            class(container_array), intent(in) this )  [private]
```

Method that returns the length of the array.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

[this]	

Definition at line 100 of file abstract_container_array.f90.

```
100     class(container_array), intent(in) :: this
101     integer :: getLength
102     getlength = this%length
```

6.2.2.2 getvalue()

```
class(*) function, pointer abstract_container_array_mod::getvalue (
            class(container_array), intent(in) this,
            integer, intent(in) index )  [private]
```

Method that returns returns the requested entry (pointer)

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

[this,index]	

Definition at line 66 of file abstract_container_array.f90.

```
66      class(container_array), intent(in) :: this
67      integer, intent(in) :: index
68      class(*), pointer :: getValue
69      if (index .le. this%getLength()) then
70          getvalue => this%contents(index)%getContent()
71      else
72          stop '[getValue]: index out of bounds'
73      endif
```

6.2.2.3 initarray()

```
subroutine abstract_container_array_mod::initarray (
            class(container_array), intent(inout) this,
            integer, intent(in) entries,
            type(container), dimension(:), intent(in), optional tocopy )  [private]
```

Method that allocates the container array. Deallocates if already allocated.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

[this,entries,tocopy]	

Definition at line 133 of file abstract_container_array.f90.

```
133     class(container_array), intent(inout) :: this
```

```
integer, intent(in) :: entries
type(container), dimension(:), optional, intent(in) :: tocopy
if (allocated(this%contents)) then
    deallocate(this%contents)
end if
if (.not.present(tocopy)) then !allocating an empty array with 'entries'
    allocate(this%contents(entries))
    this%length=entries
else if (present(tocopy)) then !using sourced allocation
  allocate(this%contents, source=tocopy)
  this%length=size(tocopy)
endif
```

6.2.2.4 putvalue()

```
subroutine abstract_container_array_mod::putvalue (
              class(container_array), intent(inout) this,
              integer, intent(in) index,
              class(*), intent(in) value )  [private]
```

Method that stores a value on the requested index.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

[this,index,value]	

Definition at line 83 of file abstract_container_array.f90.

```
83     class(container_array), intent(inout) :: this
84     integer, intent(in) :: index
85     class(*), intent(in) :: value
86     if (index .le. this%getLength()) then
87         call this%contents(index)%storeContent(value)
88     else
89         stop '[putValue]: index out of bounds'
90     endif
```

6.2.2.5 resizearray()

```
subroutine abstract_container_array_mod::resizearray (
              class(container_array), intent(inout) this,
              integer, intent(in) newsize )  [private]
```

Method that grows (adds empty space) or shrinks (discards the last entries) of the array. Use sparsely as this might get expensive for large array operations. Should think of a way to use move_alloc()

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

[this,newsize]	

Definition at line 114 of file abstract_container_array.f90.

```
114     class(container_array), intent(inout) :: this
115     integer, intent(in) :: newsize
116     integer :: i, tocopy
117     type(container), allocatable, dimension(:) :: temp
118     tocopy=min(this%getLength(),newsize)
119     allocate(temp(newsize))
120     do i=1, tocopy
121         call temp(i)%storeContent(this%get(i))
122     enddo
123     call this%init(newsize,temp)
```

6.3 blocks_mod Module Reference

Module that defines a block class and related methods. A block is a fundamental type of the model. It contains a sub-domain of the simulation bounding box, holding all entities inside that sub-domain. It maps to a domain decomposition parallelization strategy, if needed.

Data Types

- type block_class

Functions/Subroutines

- subroutine initblock (self, id, templatebox)

    method to allocate and initialize blocks and their emitters
- subroutine putsource (self, sourcetoput)

    Method to place a Source on the Block SourceArray. Checks for space and allocates more if needed. The array gets incremented by une unit at a time.
- subroutine printblock (self)

    Method to print basic info about the block.
- subroutine printdetailblock (self)

    Method to print detailed info about the block.
- subroutine, public setblocks (auto, nblk, nxi, nyi)

    routine to set the simulation blocks extents and call the block initializer
- subroutine, public allocblocks (nblk)

    routine to allocate the simulation blocks

Variables

- type(block_class), dimension(:), allocatable, public dblock

6.3.1 Detailed Description

Module that defines a block class and related methods. A block is a fundamental type of the model. It contains a sub-domain of the simulation bounding box, holding all entities inside that sub-domain. It maps to a domain decomposition parallelization strategy, if needed.

Author

Ricardo Birjukovs Canelas

6.3.2 Function/Subroutine Documentation

6.3.2.1 allocblocks()

```
subroutine, public blocks_mod::allocblocks (
             integer, intent(in) nblk )
```

routine to allocate the simulation blocks

Author

Ricardo Birjukovs Canelas - MARETEC

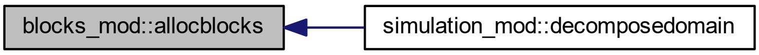
Parameters

in	*nblk*	

Definition at line 223 of file blocks.f90.

```
223     implicit none
224     integer, intent(in) ::  nblk
225     type(string) :: outext, temp
226     integer err
227     allocate(dblock(nblk), stat=err)
228     if(err/=0)then
229         outext='[allocBlobks]: Cannot allocate Blocks, stoping'
230         call log%put(outext)
231         stop
232     else
233         temp = nblk
234         outext = 'Allocated '// temp // ' Blocks.'
235         call log%put(outext)
236     endif
```

Here is the caller graph for this function:



6.3.2.2 initblock()

```
subroutine blocks_mod::initblock (
            class(block_class), intent(inout) self,
            integer, intent(in) id,
            type(box), intent(in) templatebox )  [private]
```

method to allocate and initialize blocks and their emitters

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,templatebox*	

Definition at line 69 of file blocks.f90.

```
69     implicit none
70     class(block_class), intent(inout) :: self
71     integer, intent(in) :: id
72     type(box), intent(in) :: templatebox
73     integer :: sizem
74     self%id = id
75     !setting the block sub-domain
76     self%extents%pt = templatebox%pt
77     self%extents%size = templatebox%size
78     !initializing the block emitter
79     call self%Emitter%initialize()
80     !initializing the Sources and Tracers arrays
81     call self%Source%init(1)   !Starting the Sources array with one position
82     self%Source%usedLength = 0 !But there are no stored Sources
83     call self%Tracer%init(1)   !Starting the Tracers array with one position
84     self%Tracer%usedLength = 0 !But there are no stored Tracers
85     !logging the ocupied space by the block
86     sizem = sizeof(self)
87     call simmemory%addblock(sizem)
```

6.3.2.3 printblock()

```
subroutine blocks_mod::printblock (
            class(block_class), intent(inout) self )  [private]
```

Method to print basic info about the block.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self*	

Definition at line 124 of file blocks.f90.

```
124     implicit none
125     class(block_class), intent(inout) :: self
126     type(string) :: outext, temp_str
127     temp_str = self%id
128     outext='-->Block '//temp_str//' is a'
129     call log%put(outext,.false.)
130     call geometry%print(self%extents)
131     temp_str = self%Source%usedLength
132     outext='       and has '//temp_str//' Sources'
133     call log%put(outext,.false.)
```

6.3.2.4 printdetailblock()

```
subroutine blocks_mod::printdetailblock (
            class(block_class), intent(inout) self )  [private]
```

Method to print detailed info about the block.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self*	

Definition at line 146 of file blocks.f90.

```
146     implicit none
147     class(block_class), intent(inout) :: self
148     type(string) :: outext, temp_str
149     integer :: i
150     temp_str = self%id
151     outext='-->Block '//temp_str//' is a'
152     call log%put(outext,.false.)
153     call geometry%print(self%extents)
154     temp_str = self%Source%usedLength
155     outext='       and has '//temp_str//' Sources'
156     call log%put(outext,.false.)
157     call self%Source%printArray()
```

6.3.2.5 putsource()

```
subroutine blocks_mod::putsource (
            class(block_class), intent(inout) self,
            class(source_class), intent(inout) sourcetoput )  [private]
```

Method to place a Source on the Block SourceArray. Checks for space and allocates more if needed. The array gets incremented by une unit at a time.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

`in`	*self,sourcetoput*	
`in,out`	*sourcetoput*	Source object to store

Definition at line 101 of file blocks.f90.

```
101     implicit none
102     class(block_class), intent(inout) :: self
103     class(source_class), intent(inout) :: sourcetoput
104 
105     !Check if the array is at capacity and needs to be resized
106     if (self%Source%usedLength == self%Source%getLength()) then
107         call self%Source%resize(self%Source%getLength()+1) !incrementing one entry
108     end if
109     self%Source%usedLength = self%Source%usedLength + 1
110     call self%Source%put(self%Source%usedLength, sourcetoput)
111 
```

6.3.2.6 setblocks()

```
subroutine, public blocks_mod::setblocks (
               logical, intent(in) auto,
               integer, intent(in) nblk,
               integer, intent(out) nxi,
               integer, intent(out) nyi )
```

routine to set the simulation blocks extents and call the block initializer

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

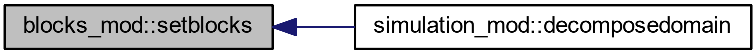
`in`	*self*	

Definition at line 171 of file blocks.f90.

```
171     implicit none
172     logical, intent(in) ::  auto
173     integer, intent(in) ::  nblk
174     integer, intent(out) :: nxi, nyi
175     type(string) :: outext, temp(2)
176     integer :: i, j, b
177     real(prec) :: ar
178     type(box) :: tempbox
179 
180     if (auto) then
181         ar = bbox%size%x/bbox%size%y
182         ar = get_closest_twopow(ar) !aspect ratio of our bounding box
183         nyi = sqrt(nblk/ar)
```

```
        if (nyi == 0) then
            temp(1) = ar
            outext='[setBlocks]: block auto sizing failed. Bouding box aspect ratio = '//temp(1)//'. Stoping'
            call log%put(outext)
            stop
        endif
        nxi = (nblk/nyi)

        b=1
        do i=1, nxi
            do j=1, nyi
              tempbox%pt = bbox%pt + bbox%size%x*(i-1)/nxi*ex + bbox%size%y*(j-1)/nyi*ey - bbox%pt%z*ez
              tempbox%size = bbox%size%x/nxi*ex + bbox%size%y/nyi*ey
              call dblock(b)%initialize(b, tempbox)
              b=b+1
            end do
        end do
        temp(1) = nxi
        temp(2) = nyi
        outext='-->Automatic domain decomposition sucessful. Domain is '//temp(1)// ' X ' //temp(2)//' Blocks'
        call log%put(outext,.false.)
    end if
    !do i=1, size(DBlock)
    !    call DBlock(i)%print()
    !enddo

    return
```

Here is the caller graph for this function:



6.3.3 Variable Documentation

6.3.3.1 dblock

`type(block_class), dimension(:), allocatable, public blocks_mod::dblock`

Definition at line 50 of file blocks.f90.

```
50      type(block_class), allocatable, dimension(:) :: DBlock
```

6.4 boundingbox_mod Module Reference

Module that defines a simulation Bounding Box.

Data Types

- type boundingbox_class

Functions/Subroutines

- subroutine initboundingbox (self)

  Method to initialize the simulation Bounding Box.
- subroutine printboundingbox (self)

  Method to print the simulation Bounding Box.

Variables

- type(boundingbox_class), public bbox

6.4.1 Detailed Description

Module that defines a simulation Bounding Box.

Author

Ricardo Birjukovs Canelas

6.4.2 Function/Subroutine Documentation

6.4.2.1 initboundingbox()

```
subroutine boundingbox_mod::initboundingbox (
            class(boundingbox_class), intent(inout) self )  [private]
```

Method to initialize the simulation Bounding Box.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 45 of file boundingbox.f90.

```
    implicit none
    class(boundingbox_class), intent(inout) :: self
    self%pt = globals%SimDefs%Pointmin
    self%size = globals%SimDefs%Pointmax - globals%SimDefs%Pointmin
    self%offset = -self%pt !distance to the origin - local reference
```

6.4.2.2 printboundingbox()

```
subroutine boundingbox_mod::printboundingbox (
               class(boundingbox_class), intent(inout) self )  [private]
```

Method to print the simulation Bounding Box.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 60 of file boundingbox.f90.

```
60      implicit none
61      class(boundingbox_class), intent(inout) :: self
62      type(string) :: outext
63      type(string) :: temp_str(3)
64
65      outext = '-->Main bounding box is '//new_line('a')
66      temp_str(1)=self%pt%x
67      temp_str(2)=self%pt%y
68      temp_str(3)=self%pt%z
69      outext = outext//'       Point = '//temp_str(1)//' '//temp_str(2)//' '//temp_str(3)//new_line('a')
70      temp_str(1)=self%size%x
71      temp_str(2)=self%size%y
72      temp_str(3)=self%size%z
73      outext = outext//'       Size = '//temp_str(1)//' '//temp_str(2)//' '//temp_str(3)
74
75      call log%put(outext,.false.)
76
```

6.4.3 Variable Documentation

6.4.3.1 bbox

```
type(boundingbox_class), public boundingbox_mod::bbox
```

Definition at line 33 of file boundingbox.f90.

```
33    type(boundingbox_class), public :: BBox
```

6.5 commom_modules Module Reference

Module to hold all of the commonly used base modules.

6.5.1 Detailed Description

Module to hold all of the commonly used base modules.

Author

Ricardo Birjukovs Canelas

6.6 container_mod Module Reference

Module that defines an unlimited polymorphic container class and related methods. A container is a fundamental entity allowing to build data structures such as lists and arrays.

Data Types

- interface container

Functions/Subroutines

- class(∗) function, pointer getcontent (this)

  Method that returns a pointer to the values stored in the container.
- subroutine storecontent (this, to_store)

  Method that stores the provided value in the container using sourced allocation.
- subroutine printcontainer (this)

  Method to print the stored value. Only knows about instrinsic types, ignores (but warns) if other types are passed.
- class(container) function, pointer constructor (to_store)

  Container constructor, can be used with the 'container' name since it is defined as an interface.

6.6.1 Detailed Description

Module that defines an unlimited polymorphic container class and related methods. A container is a fundamental entity allowing to build data structures such as lists and arrays.

Author

Ricardo Birjukovs Canelas

6.6.2 Function/Subroutine Documentation

6.6.2.1 constructor()

```
class(container) function, pointer container_mod::constructor (
            class(∗), intent(in) to_store )  [private]
```

Container constructor, can be used with the 'container' name since it is defined as an interface.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

[to_store]	

Definition at line 109 of file container.f90.

```
109     class(container), pointer :: constructor
110     class(*), intent(in) :: to_store
111     allocate(constructor)
112     allocate(constructor%value, source=to_store)
```

6.6.2.2 getcontent()

```
class(*) function, pointer container_mod::getcontent (
            class(container), intent(in) this )  [private]
```

Method that returns a pointer to the values stored in the container.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

[this]	

Definition at line 62 of file container.f90.

```
62     class(container), intent(in) :: this
63     class(*), pointer :: getContent
64     getcontent => this%value
```

6.6.2.3 printcontainer()

```
subroutine container_mod::printcontainer (
            class(container), intent(in) this )  [private]
```

Method to print the stored value. Only knows about instrinsic types, ignores (but warns) if other types are passed.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

[this]	

Definition at line 88 of file container.f90.

```
88      class(container), intent(in) :: this
89      select type(v => this%value)
90      type is (integer)
91          print *, v
92      type is (character(*))
93          print *, v(1:1)
94      type is (real)
95          print *, v
96          class default
97          print*, "[printContainer]: don't know how to print this value, ignoring"
98      end select
```

6.6.2.4 storecontent()

```
subroutine container_mod::storecontent (
            class(container), intent(inout) this,
            class(*), intent(in) to_store )  [private]
```

Method that stores the provided value in the container using sourced allocation.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

[this,to_store]	

Definition at line 75 of file container.f90.

```
75      class(container), intent(inout) :: this
76      class(*), intent(in) :: to_store
77      allocate(this%value, source=to_store)
```

6.7 emitter_mod Module Reference

Module that defines an emitter class and related methods. This module is responsible for building a potential tracer list based on the availble sources and calling their initializers.

Data Types

- type emitter_class

Functions/Subroutines

- subroutine initracers (self, srcs)

  method that calls the tracer initialization from the emmiter object
- subroutine alloctracers (self, src)

  method that allocates the tracers respective to a given source
- subroutine initializeemitter (self)

  method that initializes an emmiter class object. Sets default values
- subroutine addsource (self, src)

  method to compute the total emittable particles per source and allocate them
- subroutine setotalnp (src)

  private routine that returns the total number of tracers an input source will potentially create

6.7.1 Detailed Description

Module that defines an emitter class and related methods. This module is responsible for building a potential tracer list based on the availble sources and calling their initializers.

Author

Ricardo Birjukovs Canelas

6.7.2 Function/Subroutine Documentation

6.7.2.1 addsource()

```
subroutine emitter_mod::addsource (
            class(emitter_class), intent(inout) self,
            class(source_class), intent(inout) src )  [private]
```

method to compute the total emittable particles per source and allocate them

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

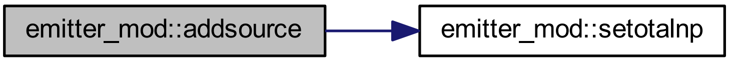
in	*self,src*	

Definition at line 141 of file emitter.f90.

```
141     implicit none
142     class(emitter_class), intent(inout) :: self
143     class(source_class),intent(inout) :: src
144     integer :: i
145
```

```
    call setotalnp(src) !finding the total tracers this Source will pass the emmiter
    self%emittable = self%emittable + src%stencil%total_np
        !print*, srcs(i)%stencil%total_np

    !allocating and initializing the tracers by the emitter, for all sources
    call self%alloctracers(src)
    !call self%initracers(srcs)

```

Here is the call graph for this function:



6.7.2.2 alloctracers()

```
subroutine emitter_mod::alloctracers (
              class(emitter_class), intent(inout) self,
              class(source_class), intent(inout) src )  [private]
```

method that allocates the tracers respective to a given source

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,src*	

Definition at line 88 of file emitter.f90.

```
    implicit none
    class(emitter_class), intent(inout) :: self
    class(source_class), intent(inout) :: src
    integer err
    type(string) :: outext, temp

    if (self%emittable .le. 0) then
        outext='[Emitter::alloctracers]: No Tracers will be simulated, stoping'
        call log%put(outext)
        stop
    else
        allocate(tracer(self%emittable), stat=err)
         if(err/=0)then
             outext='[Emitter::alloctracers]: Cannot allocate Tracers, stoping'
             call log%put(outext)
             stop
         endif
     endif

```

```
107     temp = size(tracer)
108     outext='Allocated '// temp // ' Tracers.'
109     call log%put(outext)
110     !receiving Sources as argument so latter we can differentiate between tracer types
111
```

6.7.2.3 initializeemitter()

```
subroutine emitter_mod::initializeemitter (
            class(emitter_class), intent(inout) self )  [private]
```

method that initializes an emmiter class object. Sets default values

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self*	

Definition at line 124 of file emitter.f90.

```
124     implicit none
125     class(emitter_class), intent(inout) :: self
126     self%emitted = 0
127     self%emittable = 0
```

6.7.2.4 initracers()

```
subroutine emitter_mod::initracers (
            class(emitter_class), intent(inout) self,
            class(source_class), dimension(:), intent(inout) srcs )  [private]
```

method that calls the tracer initialization from the emmiter object

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,src*	

Definition at line 56 of file emitter.f90.

```
    implicit none
    class(emitter_class), intent(inout) :: self
    class(source_class), dimension(:), intent(inout) :: srcs
    integer num_emiss, i, j, k, p
    type(string) :: outext, temp(4)
    integer :: sizem

    p=0
    do i=1, size(srcs)
        num_emiss = srcs(i)%stencil%total_np/size(srcs(i)%stencil%ptlist)
        do j=1, num_emiss
            do k=1, size(srcs(i)%stencil%ptlist)
                p=p+1
                call tracer(p)%initialize(p, srcs(i)%par%id, globals%SimTime, srcs(i)%stencil%ptlist(k))
            enddo
        enddo
    enddo
    sizem = sizeof(tracer)
    call simmemory%addtracer(sizem)

```

6.7.2.5 setotalnp()

```
subroutine emitter_mod::setotalnp (
              class(source_class), intent(inout) src )   [private]
```

private routine that returns the total number of tracers an input source will potentially create

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

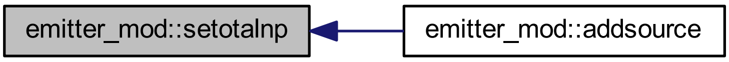
in	*src*	

$$NP^{source-i}_{total} = (T^{source-i}_{end} - T^{source-i}_{start}) * Rate^{source-i} * NP^{source-i}_{emission}$$

Definition at line 167 of file emitter.f90.

```
167     implicit none
168     class(source_class), intent(inout) :: src
170     src%stencil%total_np=(src%par%stoptime-src%par%startime)*src%par%emitting_rate*src%stencil%np
```

Here is the caller graph for this function:



6.8 geometry_mod Module Reference

Module that defines geometry classes and related methods.

Data Types

- type box

  Type - point class.
- type geometry_class
- type line

  Type - line class.
- type point

  Type - point class.
- type shape

  Type - extendable shape class.
- type sphere

  Type - sphere class.

Functions/Subroutines

- subroutine allocatelist (self)

  Public routine to allocate the possible geometry name list.
- logical function inlist (self, geomname)

  Public function that returns a logical if the input geometry name is valid.
- integer function fillsize (self, shapetype)

  method to get the number of points that fill a given geometry
- subroutine fill (self, shapetype, fillsize, ptlist)

  method to get the list of points that fill a given geometry
- subroutine printgeometry (self, shapetype)

  method to print the details of a given geometry
- integer function sphere_np_count (dp, r)

  private function that returns the number of points distributed on a grid with spacing dp inside a sphere
- subroutine sphere_grid (dp, r, np, ptlist)

  private routine that returns the points distributed on a grid with spacing dp inside a sphere
- subroutine box_grid (dp, size, np, ptlist)

  private routine that returns the points distributed on a grid with spacing dp inside a box
- subroutine line_grid (dp, dist, np, ptlist)

  private routine that returns the points distributed on a grid with spacing dp along a line

Variables

- type(geometry_class), public geometry

6.8.1 Detailed Description

Module that defines geometry classes and related methods.

Author

Ricardo Birjukovs Canelas

6.8.2 Function/Subroutine Documentation

6.8.2.1 allocatelist()

```
subroutine geometry_mod::allocatelist (
            class(geometry_class), intent(inout) self )  [private]
```

Public routine to allocate the possible geometry name list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 77 of file geometry.f90.

```
implicit none
class(geometry_class), intent(inout) :: self
allocate(self%list(4))
self%list(1) ='point'
self%list(2) ='line'
self%list(3) ='box'
self%list(4) ='sphere'
```

6.8.2.2 box_grid()

```
subroutine geometry_mod::box_grid (
            real(prec), intent(in) dp,
            type(vector), intent(in) size,
            integer, intent(in) np,
            type(vector), dimension(np), intent(out) ptlist )  [private]
```

private routine that returns the points distributed on a grid with spacing dp inside a box

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

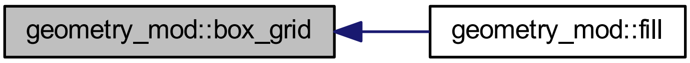
in	*dp,size,np,ptlist*	

Definition at line 316 of file geometry.f90.

```
implicit none
real(prec), intent(in) :: dp
type(vector), intent(in) :: size
integer, intent(in)::  np
```

```
    type(vector), intent(out) :: ptlist(np)
    integer :: i, j, k, p
    p=0
    do i=1, int(size%x/dp)+1
        do j=1, int(size%y/dp)+1
            do k=1, int(size%z/dp)+1
                p=p+1
                ptlist(p) = dp*(ex*(i-1)+ey*(j-1)+ez*(k-1))
            end do
        end do
    end do
    if (np == 1) then !Just the origin
        ptlist(1)= 0*ex + 0*ey +0*ez
    end if
```

Here is the caller graph for this function:



6.8.2.3 fill()

```
subroutine geometry_mod::fill (
            class(geometry_class), intent(in) self,
            class(shape) shapetype,
            integer, intent(in) fillsize,
            type(vector), dimension(fillsize), intent(out) ptlist )  [private]
```

method to get the list of points that fill a given geometry

Author

Ricardo Birjukovs Canelas - MARETEC

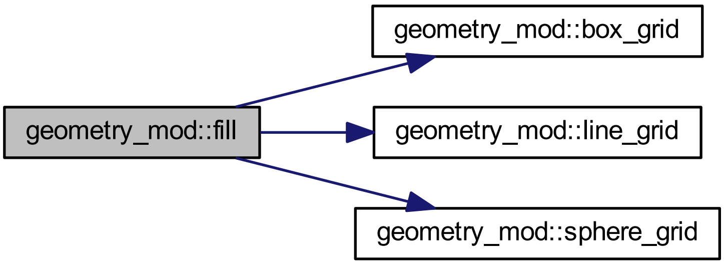
Parameters

in	*shapetype,fillsize,ptlist*	

Definition at line 156 of file geometry.f90.

```
    implicit none
    class(geometry_class), intent(in) :: self
    class(shape) :: shapetype
    integer, intent(in) :: fillsize
    type(vector), intent(out) :: ptlist(fillsize)
    type(vector) :: temp
    type(string) :: outext

    select type (shapetype)
    type is (shape)
    class is (box)
        call box_grid(globals%SimDefs%Dp, shapetype%size, fillsize, ptlist)
    class is (point)
        ptlist(1)=shapetype%pt
    class is (line)
        call line_grid(globals%SimDefs%Dp, shapetype%last-shapetype%pt, fillsize, ptlist)
    class is (sphere)
        call sphere_grid(globals%SimDefs%Dp, shapetype%radius, fillsize, ptlist)
        class default
        outext='[geometry::fill] : unexpected type for geometry object, stoping'
        call log%put(outext)
        stop
    end select

```

Here is the call graph for this function:



6.8.2.4 fillsize()

```
integer function geometry_mod::fillsize (
            class(geometry_class), intent(in) self,
            class(shape), intent(in) shapetype )  [private]
```

method to get the number of points that fill a given geometry

Author

Ricardo Birjukovs Canelas - MARETEC

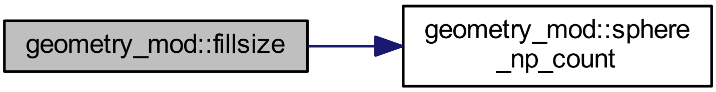
Parameters

in	*shapetype*	

Definition at line 118 of file geometry.f90.

```
    implicit none
    class(geometry_class), intent(in) :: self
    class(shape), intent(in) :: shapetype
    real(prec) :: dp
    integer :: fillsize
    type(vector) :: temp
    type(string) :: outext

    dp = globals%SimDefs%Dp
    select type (shapetype)
    type is (shape)
    class is (box)
        fillsize = max((int(shapetype%size%x/dp)+1)*(int(shapetype%size%y/dp)+1)*(int(shapetype%size%z/dp)+
  1),1)
    class is (point)
        fillsize = 1
    class is (line)
        temp = shapetype%pt-shapetype%last
        fillsize = max(int(temp%normL2()/dp),1)
    class is (sphere)
        fillsize = sphere_np_count(dp, shapetype%radius)
        class default
        outext='[geometry::np] : unexpected type for geometry object, stoping'
        call log%put(outext)
        stop
    end select

```

Here is the call graph for this function:



6.8.2.5 inlist()

```
logical function geometry_mod::inlist (
              class(geometry_class), intent(in) self,
              type(string), intent(in) geomname )  [private]
```

Public function that returns a logical if the input geometry name is valid.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*geomname*	

Definition at line 96 of file geometry.f90.

```
96    implicit none
```

```
class(geometry_class), intent(in) :: self
type(string), intent(in) :: geomname
integer :: i
 tf = .false.
 do i=1, size(self%list)
     if (geomname == self%list(i)) then
         tf = .true.
     endif
 enddo
```

6.8.2.6 line_grid()

```
subroutine geometry_mod::line_grid (
               real(prec), intent(in) dp,
               type(vector), intent(in) dist,
               integer, intent(in) np,
               type(vector), dimension(np), intent(out) ptlist ) [private]
```

private routine that returns the points distributed on a grid with spacing dp along a line

Author

Ricardo Birjukovs Canelas - MARETEC

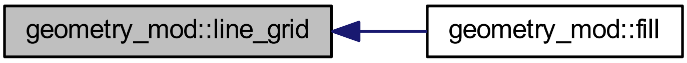
Parameters

in	*dp,size,np,ptlist*	

Definition at line 347 of file geometry.f90.

```
347     implicit none
348     real(prec), intent(in) :: dp
349     type(vector), intent(in) :: dist
350     integer, intent(in)::  np
351     type(vector), intent(out) :: ptlist(np)
352     integer :: i, j, k, p
353
354     do p=1, np
355         ptlist(p) = dp/np*(dist*(p-1))
356     end do
357     if (np == 1) then !Just the origin
358         ptlist(1)= 0*ex + 0*ey +0*ez
359     end if
```

Here is the caller graph for this function:



6.8.2.7 printgeometry()

```
subroutine geometry_mod::printgeometry (
            class(geometry_class), intent(in) self,
            class(shape) shapetype )  [private]
```

method to print the details of a given geometry

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*shapetype*	

Definition at line 191 of file geometry.f90.

```
implicit none
class(geometry_class), intent(in) :: self
class(shape) :: shapetype

type(vector) :: temp(2)
type(string) :: temp_str(6)
type(string) :: outext

temp_str(1) = shapetype%pt%x
temp_str(2) = shapetype%pt%y
temp_str(3) = shapetype%pt%z
select type (shapetype)
type is (shape)
class is (box)
    temp_str(4) = shapetype%size%x
    temp_str(5) = shapetype%size%y
    temp_str(6) = shapetype%size%z
    outext='      Box at '//temp_str(1)//' '//temp_str(2)//' '//temp_str(3)//new_line('a')//&
           '       with '//temp_str(4)//' X '//temp_str(5)//' X '//temp_str(6)
class is (point)
    outext='      Point at '//temp_str(1)//' '//temp_str(2)//' '//temp_str(3)
class is (line)
    temp_str(4) = shapetype%last%x
    temp_str(5) = shapetype%last%y
    temp_str(6) = shapetype%last%z
    outext='      Line from '//temp_str(1)//' '//temp_str(2)//' '//temp_str(3)//new_line('a')//&
           '       to '//temp_str(4)//' X '//temp_str(5)//' X '//temp_str(6)
class is (sphere)
    temp_str(4) = shapetype%radius
    outext='      Sphere at '//temp_str(1)//' '//temp_str(2)//' '//temp_str(3)//new_line('a')//&
           '       with radius '//temp_str(4)
    class default
    outext='[geometry::print] : unexpected type for geometry object, stoping'
    call log%put(outext)
    stop
end select
call log%put(outext,.false.)

```

6.8.2.8 sphere_grid()

```
subroutine geometry_mod::sphere_grid (
            real(prec), intent(in) dp,
            real(prec), intent(in) r,
            integer, intent(in) np,
            type(vector), dimension(np), intent(out) ptlist )  [private]
```

private routine that returns the points distributed on a grid with spacing dp inside a sphere

Author

Ricardo Birjukovs Canelas - MARETEC

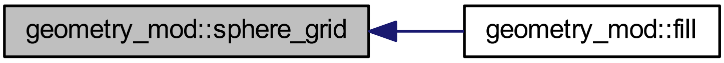
Parameters

in	*dp,r,np,ptlist*	

Definition at line 278 of file geometry.f90.

```
278     implicit none
279     real(prec), intent(in) :: dp
280     real(prec), intent(in) :: r
281     integer, intent(in)::  np
282     type(vector), intent(out) :: ptlist(np)
283     integer :: i, j, k, p, n
284     type(vector) :: pts
285     n=int(3*r/dp)
286     p=0
287     do i=1, n
288         do j=1, n
289             do k=1, n
290                 pts = dp*(ex*(i-1)+ey*(j-1)+ez*(k-1)) - r*(ex+ey+ez)
291                 if (pts%normL2() .le. r) then
292                     p=p+1
293                     ptlist(p)=pts
294                 end if
295             end do
296         end do
297     end do
298     if (np == 1) then !Just the center point
299         ptlist(1)= 0*ex + 0*ey +0*ez
300     end if
301
```

Here is the caller graph for this function:



6.8.2.9 sphere_np_count()

```
integer function geometry_mod::sphere_np_count (
            real(prec), intent(in) dp,
            real(prec), intent(in) r )  [private]
```

private function that returns the number of points distributed on a grid with spacing dp inside a sphere

Author

Ricardo Birjukovs Canelas - MARETEC

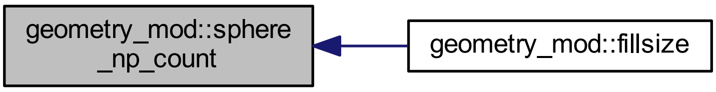
Parameters

in	*dp,r*	

Definition at line 243 of file geometry.f90.

```
implicit none
real(prec), intent(in) :: dp
real(prec), intent(in) :: r
integer :: np
integer :: i, j, k, n
type(vector) :: pts
np=0
n=int(3*r/dp)
do i=1, n
    do j=1, n
        do k=1, n
            pts = dp*(ex*(i-1)+ey*(j-1)+ez*(k-1)) - r*(ex+ey+ez)
            if (pts%normL2() .le. r) then
                np=np+1
            end if
        end do
    end do
end do
if (np == 0) then !Just the center point
    np=1
end if

```

Here is the caller graph for this function:



6.8.3 Variable Documentation

6.8.3.1 geometry

```
type(geometry_class), public geometry_mod::geometry
```

Definition at line 61 of file geometry.f90.

```
61     type(geometry_class) :: Geometry
```

6.9 initialize_mod Module Reference

Module with the simulation initialization related definitions and methods. Has one public access routine that is incharge of building the simulation space from input files.

Functions/Subroutines

- subroutine linkpropertysources (linksNode)

  Private property xml parser routine. Reads the properties tab from the xml file and links these to the corresponding source.
- subroutine init_properties (case_node)

  Private property xml parser routine. Reads the properties tab from the xml file and links these to the corresponding source.
- subroutine read_xml_geometry (source, source_detail, source_shape)

  Private geometry xml parser routine. Reads a geometry from the xml depending on the geometry type of the node.
- subroutine init_sources (case_node)

  Private source definitions parser routine. Builds the tracer sources from the input xml case file.
- subroutine init_simdefs (case_node)

  Private simulation definitions parser routine. Builds the simulation geometric space from the input xml case file.
- subroutine init_caseconstants (case_node)

  Private case constant parser routine. Builds the simulation parametric space from the input xml case file.
- subroutine init_parameters (execution_node)

  Private parameter parser routine. Builds the simulation parametric space from the input xml case file.
- subroutine, public initfromxml (xmlfilename)

  Public xml parser routine. Builds the simulation space from the input xml case file.

6.9.1 Detailed Description

Module with the simulation initialization related definitions and methods. Has one public access routine that is incharge of building the simulation space from input files.

Author

Ricardo Birjukovs Canelas

6.9.2 Function/Subroutine Documentation

6.9.2.1 init_caseconstants()

```
subroutine initialize_mod::init_caseconstants (
            type(node), intent(in), pointer case_node )  [private]
```

Private case constant parser routine. Builds the simulation parametric space from the input xml case file.

Author

Ricardo Birjukovs Canelas - MARETEC

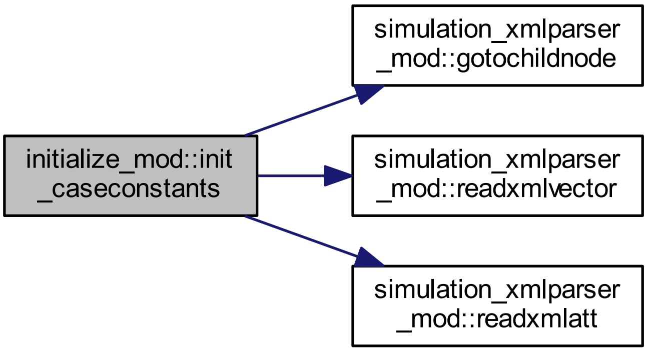
Parameters

in	*case_node*	

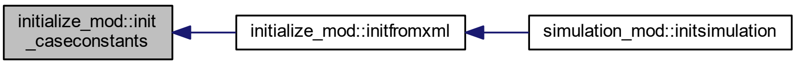
Definition at line 305 of file initialize.f90.

```
    implicit none
    type(Node), intent(in), pointer :: case_node

    type(Node), pointer :: constants_node
    type(string) :: outext
    type(string) :: tag, att_name, att_val
    type(vector) :: coords
    logical :: readflag

    outext='-->Reading case constants'
    call log%put(outext,.false.)

    tag="constantsdef"    !the node we want
    call gotochildnode(case_node,constants_node,tag,readflag,.false.)
    if (readflag) then !if the node exists, since his one is not mandatory
      tag="Gravity"
      call readxmlvector(constants_node,tag,coords,readflag,.false.)
      if (readflag) then
        call globals%Constants%setgravity(coords)
      endif
      tag="Z0"
      att_name="value"
      call readxmlatt(constants_node, tag, att_name, att_val,readflag,.false.)
      if (readflag) then
        call globals%Constants%setz0(att_val)
      endif
      tag="Rho_ref"
      att_name="value"
      call readxmlatt(constants_node, tag, att_name, att_val,readflag,.false.)
      if (readflag) then
        call globals%Constants%setrho(att_val)
      endif
    endif
    call globals%Constants%print()

```

Here is the call graph for this function:


initialize_mod::init _caseconstants
simulation_xmlparser _mod::gotochildnode
simulation_xmlparser _mod::readxmlvector
simulation_xmlparser _mod::readxmlatt

Here is the caller graph for this function:


initialize_mod::init _caseconstants
initialize_mod::initfromxml
simulation_mod::initsimulation

6.9.2.2 init_parameters()

```
subroutine initialize_mod::init_parameters (
            type(node), intent(in), pointer execution_node )  [private]
```

Private parameter parser routine. Builds the simulation parametric space from the input xml case file.

Author

Ricardo Birjukovs Canelas - MARETEC

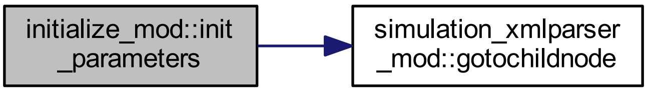
Parameters

in	*execution_node*	

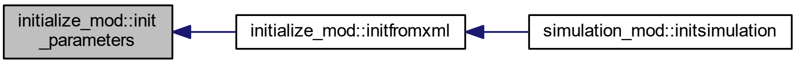
Definition at line 352 of file initialize.f90.

```
352     implicit none
353     type(Node), intent(in), pointer :: execution_node
354 
355     type(string) :: outext
356     type(NodeList), pointer :: parameterList
357     type(Node), pointer :: parmt, parameters_node
358     integer :: i
359     type(string) :: parmkey, parmvalue, tag
360     character(80) :: parmkey_char, parmvalue_char
361 
362     outext='-->Reading case parameters'
363     call log%put(outext,.false.)
364 
365     tag="parameters"    !the node we want
366     call gotochildnode(execution_node,parameters_node,tag)
367     parameterlist => getelementsbytagname(parameters_node, "parameter")       !searching for tags with the
       'parameter' name
368     do i = 0, getlength(parameterlist) - 1                          !extracting parameter tags one by one
369         parmt => item(parameterlist, i)
370         call extractdataattribute(parmt, "key", parmkey_char)       !name of the parameter
371         call extractdataattribute(parmt, "value", parmvalue_char)   !value of the parameter
372         parmkey=trim(parmkey_char)
373         parmvalue=trim(parmvalue_char)
374         call globals%Parameters%setparameter(parmkey,parmvalue)
375     enddo
376     call globals%Parameters%check()
377     call globals%Parameters%print()
378 
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.9.2.3 init_properties()

```
subroutine initialize_mod::init_properties (
            type(node), intent(in), pointer case_node )  [private]
```

Private property xml parser routine. Reads the properties tab from the xml file and links these to the corresponding source.

Author

Ricardo Birjukovs Canelas - MARETEC

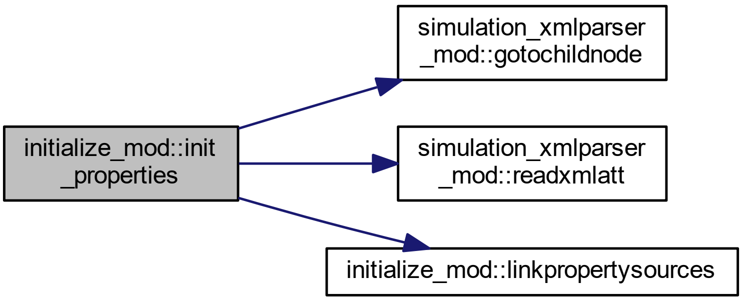
Parameters

in	*parsedxml*	

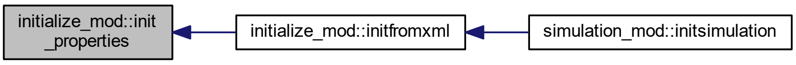
Definition at line 85 of file initialize.f90.

```
85      implicit none
86      type(Node), intent(in), pointer :: case_node
87
88      type(Node), pointer :: props_node
89      type(string) :: outext
90      type(string) :: tag, att_name
91
92      tag="properties"    !the node we want
93      call gotochildnode(case_node,props_node,tag)
94      if (associated(props_node)) then
95          tag="propertyfile"
96          att_name="name"
97          call readxmlatt(props_node, tag, att_name, globals%FileNames%propsxmlfilename)  !getting the file
        name from that tag
98          outext='-->Properties to link to Sources found at '//globals%FileNames%propsxmlfilename
99          call log%put(outext,.false.)
100          tag="links"
101          call gotochildnode(props_node,props_node,tag) !getting the links node
102          call linkpropertysources(props_node)          !calling the property linker
103      else
104          outext='-->No properties to link to Sources, assuming pure Lagrangian tracers'
105          call log%put(outext,.false.)
106      endif
107
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.9.2.4 init_simdefs()

```
subroutine initialize_mod::init_simdefs (
            type(node), intent(in), pointer case_node )  [private]
```

Private simulation definitions parser routine. Builds the simulation geometric space from the input xml case file.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

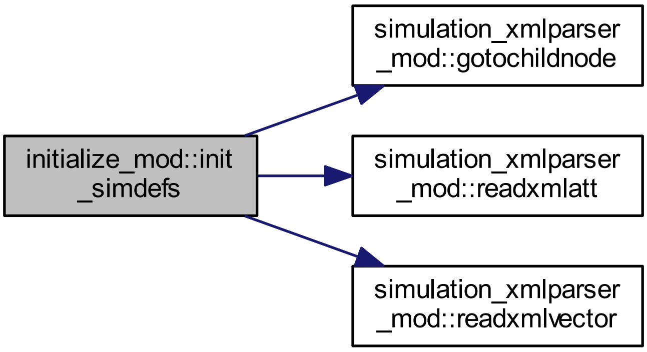
in	*case_node*	

Definition at line 263 of file initialize.f90.

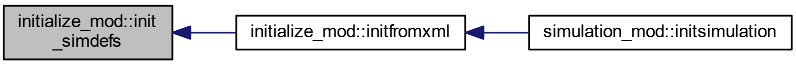
```
    implicit none
    type(Node), intent(in), pointer :: case_node

    type(NodeList), pointer :: defsList
```

```
type(Node), pointer :: simdefs_node
type(string) :: outext
integer :: i
type(string) :: pts(2), tag, att_name, att_val
type(vector) :: coords

outext='-->Reading case simulation definitions'
call log%put(outext,.false.)

tag="simulationdefs"    !the node we want
call gotochildnode(case_node,simdefs_node,tag)
tag="resolution"
att_name="dp"
call readxmlatt(simdefs_node, tag, att_name, att_val)
call globals%SimDefs%setdp(att_val)
tag="timestep"
att_name="dt"
call readxmlatt(simdefs_node, tag, att_name, att_val)
call globals%SimDefs%setdt(att_val)
pts=(/ 'pointmin', 'pointmax'/) !strings to search for
do i=1, size(pts)
    call readxmlvector(simdefs_node, pts(i), coords)
    call globals%SimDefs%setboundingbox(pts(i), coords)
enddo
call globals%SimDefs%print()
```

Here is the call graph for this function:


initialize_mod::init _simdefs
simulation_xmlparser _mod::gotochildnode
simulation_xmlparser _mod::readxmlatt
simulation_xmlparser _mod::readxmlvector

Here is the caller graph for this function:


initialize_mod::init _simdefs
initialize_mod::initfromxml
simulation_mod::initsimulation

6.9.2.5 init_sources()

```
subroutine initialize_mod::init_sources (
            type(node), intent(in), pointer case_node )  [private]
```

Private source definitions parser routine. Builds the tracer sources from the input xml case file.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

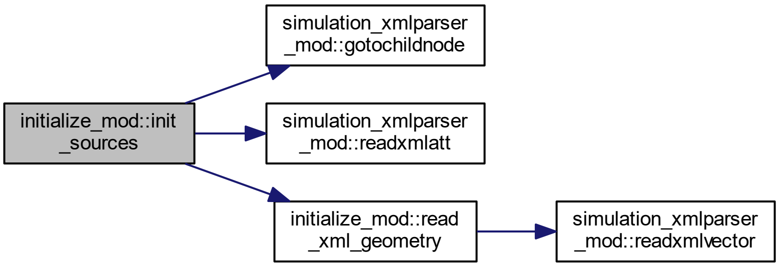
in	*case_node*	

Definition at line 167 of file initialize.f90.

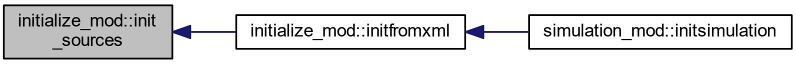
```
167     implicit none
168     type(Node), intent(in), pointer :: case_node
169 
170     type(string) :: outext
171     type(NodeList), pointer :: sourceList
172     type(NodeList), pointer :: sourceChildren
173     type(Node), pointer :: sourcedef
174     type(Node), pointer :: source_node
175     type(Node), pointer :: source_detail
176     integer :: i, j
177     logical :: readflag
178     !source vars
179     integer :: id
180     type(string) :: name, source_geometry, tag, att_name, att_val
181     real(prec) :: emitting_rate, start, finish
182     class(shape), allocatable :: source_shape
183 
184     outext='-->Reading case Sources'
185     call log%put(outext,.false.)
186 
187     tag="sourcedef"    !the node we want
188     call gotochildnode(case_node,sourcedef,tag)
189     sourcelist => getelementsbytagname(sourcedef, "source")
190 
191     !allocating the temporary source objects
192     call tempsources%initialize(getlength(sourcelist))
193 
194     do j = 0, getlength(sourcelist) - 1
195         source_node => item(sourcelist,j)
196         tag="setsource"
197         att_name="id"
198         call readxmlatt(source_node, tag, att_name, att_val)
199         id=att_val%to_number(kind=1_i1p)
200         att_name="name"
201         call readxmlatt(source_node, tag, att_name, name)
202         tag="set"
203         att_name="emitting_rate"
204         call readxmlatt(source_node, tag, att_name, att_val)
205         emitting_rate = att_val%to_number(kind=1._r4p)
206         tag="active"
207         att_name="start"
208         call readxmlatt(source_node, tag, att_name, att_val,readflag,.false.)
209         if (readflag) then
210             start = att_val%to_number(kind=1._r4p)
211         else
212             start = 0.0
213         endif
214         att_name="end"
215         call readxmlatt(source_node, tag, att_name, att_val,readflag,.false.)
216         if (readflag.and.att_val%is_number()) then
217             finish = att_val%to_number(kind=1._r4p)
218         else
219             finish = globals%Parameters%TimeMax
220         endif
221         !now we need to find out the geometry of the source and read accordingly
222         sourcechildren => getchildnodes(source_node) !getting all of the nodes bellow the main source node
       (all of it's private info)
```

```
      do i=0, getlength(sourcechildren)-1
          source_detail => item(sourcechildren,i) !grabing a node
          source_geometry = getlocalname(source_detail)  !finding its name
          if (geometry%inlist(source_geometry)) then  !if the node is a valid geometry name
              select case (source_geometry%chars())
              case ('point')
                  allocate(point::source_shape)
              case ('sphere')
                  allocate(sphere::source_shape)
              case ('box')
                  allocate(box::source_shape)
              case ('line')
                  allocate(line::source_shape)
              case default
                  outext='[init_sources]: unexpected type for geometry object!'
                  call log%put(outext)
                  stop
              end select
              call read_xml_geometry(source_node,source_detail,source_shape)
              exit
          endif
      enddo
      !initializing Source j
      call tempsources%src(j+1)%initialize(id,name,emitting_rate,start,finish,source_geometry,
source_shape)

      deallocate(source_shape)
  enddo

```

Here is the call graph for this function:


simulation_xmlparser _mod::gotochildnode
initialize_mod::init _sources
simulation_xmlparser _mod::readxmlatt
initialize_mod::read _xml_geometry
simulation_xmlparser _mod::readxmlvector

Here is the caller graph for this function:


initialize_mod::init _sources
initialize_mod::initfromxml
simulation_mod::initsimulation

6.9.2.6 initfromxml()

```
subroutine, public initialize_mod::initfromxml (
              type(string), intent(in) xmlfilename )
```

Public xml parser routine. Builds the simulation space from the input xml case file.

Author

Ricardo Birjukovs Canelas - MARETEC

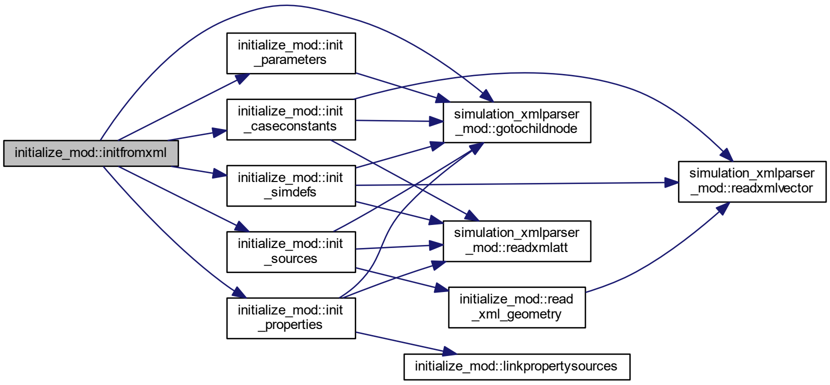
Parameters

in	*xmlfilename*	
in	*xmlfilename*	.xml file name

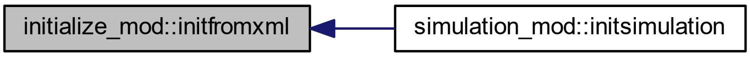
Definition at line 392 of file initialize.f90.

```
392     implicit none
393     type(string), intent(in) :: xmlfilename
394     type(string) :: outext, tag
395     type(Node), pointer :: xmldoc
396     type(Node), pointer :: case_node
397     type(Node), pointer :: execution_node
398     integer :: i
399
400     xmldoc => parsefile(xmlfilename%chars(), iostat=i)
401     if (i==0) then
402         outext='->Reading case definition from '//xmlfilename
403         call log%put(outext)
404         globals%FileNames%mainxmlfilename = xmlfilename
405     else
406         outext='[initMohidLagrangian]: no '//xmlfilename//' input file, give me at least that!'
407         call log%put(outext)
408         stop
409     endif
410
411     tag="case"          !base document node
412     call gotochildnode(xmldoc,execution_node,tag)
413     tag="execution"     !finding execution node
414     call gotochildnode(execution_node,execution_node,tag)
415     tag="case"          !base document node
416     call gotochildnode(xmldoc,case_node,tag)
417     tag="casedef"     !finding execution node
418     call gotochildnode(case_node,case_node,tag)
419
420     ! building the simulation basic structures according to the case definition file
421     ! every other structure in the simulation is built from these, i.e., not defined by the user directly
422     call init_parameters(execution_node)
423     call init_caseconstants(case_node)
424     call init_simdefs(case_node)
425     call init_sources(case_node)
426     call init_properties(case_node)
427
428     call destroy(xmldoc)
429
```

Here is the call graph for this function:


initialize_mod::initfromxml
initialize_mod::init_parameters
initialize_mod::init_caseconstants
initialize_mod::init_simdefs
initialize_mod::init_sources
initialize_mod::init_properties
simulation_xmlparser_mod::gotochildnode
simulation_xmlparser_mod::readxmlvector
simulation_xmlparser_mod::readxmlatt
initialize_mod::read_xml_geometry
initialize_mod::linkpropertysources

Here is the caller graph for this function:



6.9.2.7 linkpropertysources()

```
subroutine initialize_mod::linkpropertysources (
            type(node), intent(in), pointer linksNode )  [private]
```

Private property xml parser routine. Reads the properties tab from the xml file and links these to the corresponding source.

Author

Ricardo Birjukovs Canelas - MARETEC

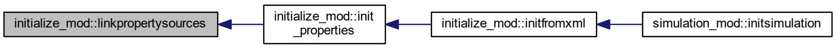
Parameters

in	*parsedxml*	

Definition at line 50 of file initialize.f90.

```
50      implicit none
51      type(Node), intent(in), pointer :: linksNode
52
53      type(NodeList), pointer :: linkList
54      type(Node), pointer :: linknode
55      integer :: i
56      character(80) :: sourceid_char, sourcetype_char, sourceprop_char
57      type(string) :: sourceid, sourcetype, sourceprop
58
59      linklist => getelementsbytagname(linksnode, "link")
60      do i = 0, getlength(linklist) - 1
61          linknode => item(linklist,i)
62          call extractdataattribute(linknode, "source", sourceid_char)
63          call extractdataattribute(linknode, "type", sourcetype_char)
64          call extractdataattribute(linknode, "property", sourceprop_char)
65          sourceid=trim(sourceid_char)
66          sourcetype=trim(sourcetype_char)
67          sourceprop=trim(sourceprop_char)
68          call tempsources%setProps(sourceid,sourcetype,sourceprop)
69      enddo
70
```

Here is the caller graph for this function:


initialize_mod::linkpropertysources
initialize_mod::init_properties
initialize_mod::initfromxml
simulation_mod::initsimulation

6.9.2.8 read_xml_geometry()

```
subroutine initialize_mod::read_xml_geometry (
            type(node), intent(in), pointer source,
            type(node), intent(in), pointer source_detail,
            class(shape), intent(inout) source_shape )  [private]
```

Private geometry xml parser routine. Reads a geometry from the xml depending on the geometry type of the node.

Author

Ricardo Birjukovs Canelas - MARETEC

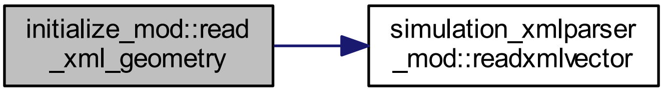
Parameters

in	*source,source_detail,source_shape*	
in	*source*	Working xml node
in	*source_detail*	Working xml node details
in,out	*source_shape*	Geometrical object to fill

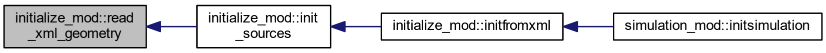
Definition at line 121 of file initialize.f90.

```
    implicit none
    type(Node), intent(in), pointer :: source
    type(Node), intent(in), pointer :: source_detail
    class(shape), intent(inout) :: source_shape
    type(string) :: outext
    type(string) :: tag

    select type (source_shape)
    type is (shape)
        !nothing to do
    class is (box)
        tag='point'
        call readxmlvector(source_detail,tag,source_shape%pt)
        tag='size'
        call readxmlvector(source_detail,tag,source_shape%size)
    class is (point)
        tag='point'
        call readxmlvector(source,tag,source_shape%pt)
    class is (line)
        tag='pointa'
        call readxmlvector(source_detail,tag,source_shape%pt)
        tag='pointb'
        call readxmlvector(source_detail,tag,source_shape%last)
    class is (sphere)
        tag='point'
        call readxmlvector(source_detail,tag,source_shape%pt)
        call extractdataattribute(source_detail, "radius", source_shape%radius)
        class default
        outext='[read_xml_geometry]: unexpected type for geometry object!'
        call log%put(outext)
        stop
    end select

```

Here is the call graph for this function:



Here is the caller graph for this function:



6.10 simulation_globals_mod Module Reference

Module to hold the simulation global parameter classes and their methods.

Data Types

- type constants_t

  Case Constants class.
- type filenames_t

  File names class.
- type globals_class

  Globals class - This is a container for every global variable on the simulation.
- type parameters_t
- type simdefs_t

  Simulation definitions class.

Functions/Subroutines

- subroutine setdefaults (self)

  Globals default setting routine.
- subroutine setparameter (self, parmkey, parmvalue)

  Private parameter setting method. Builds the simulation parametric space from the input case file.
- subroutine check (self)

  Parameter checking method. Checks if mandatory parameters were set.
- subroutine printsimparameters (self)

  Parameter printing method.
- subroutine getintegratorname (name, code)

Routine to get integrator scheme name.

- subroutine setgravity (self, grav)

  Gravity setting routine.
- subroutine setz0 (self, read_z0)

  Z0 setting routine.
- subroutine setrho (self, read_rho)

  Rho_Ref setting routine.
- subroutine printconstants (self)

  Public constants printing routine.
- subroutine setdp (self, read_dp)

  Dp setting routine.
- subroutine setdt (self, read_dt)

  Dt setting routine.
- subroutine setboundingbox (self, point_, coords)

  Bounding box setting routine.
- subroutine setblocksize (self, bsize)

  blocksize box setting routine.
- subroutine printsimdefs (self)

  Public simulation definitions printing routine.

Variables

- type(globals_class), public globals

6.10.1 Detailed Description

Module to hold the simulation global parameter classes and their methods.

Author

Ricardo Birjukovs Canelas

6.10.2 Function/Subroutine Documentation

6.10.2.1 check()

```
subroutine simulation_globals_mod::check (
            class(parameters_t), intent(inout) self )  [private]
```

Parameter checking method. Checks if mandatory parameters were set.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 183 of file simulation_globals.f90.

```
183     implicit none
184     class(parameters_t), intent(inout) :: self
185     type(string) :: outext
186 
187     !add new parameters to this search
188     if (self%TimeMax==mv) then
189         outext = 'Maximum simulation time parameter (TimeMax) is not set, stoping'
190         call log%put(outext)
191         stop
192     elseif (self%TimeOut==mv) then
193         outext = 'Simulation sampling rate parameter (TimeOut) is not set, stoping'
194         call log%put(outext)
195         stop
196     endif
```

6.10.2.2 getintegratorname()

```
subroutine simulation_globals_mod::getintegratorname (
              type(string), intent(inout) name,
              integer, intent(in) code )  [private]
```

Routine to get integrator scheme name.

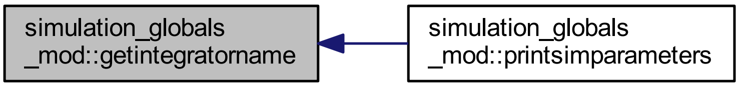
Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 232 of file simulation_globals.f90.

```
232     implicit none
233     type(string), intent(inout) :: name
234     integer, intent(in) :: code
235     if (code==1) then
236         name='Verlet'
237     elseif(code==2)then
238         name='Symplectic'
239     elseif(code==3)then
240         name='Runge-Kuta 4'
241     endif
```

Here is the caller graph for this function:



6.10.2.3 printconstants()

```
subroutine simulation_globals_mod::printconstants (
            class(constants_t), intent(in) self )  [private]
```

Public constants printing routine.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 322 of file simulation_globals.f90.

```
322     implicit none
323     class(constants_t), intent(in) :: self
324     type(string) :: outext
325     type(string) :: temp_str(3)
326 
327     temp_str(1)=self%Gravity%x
328     temp_str(2)=self%Gravity%y
329     temp_str(3)=self%Gravity%z
330     outext = '       Gravity is '//new_line('a')//&
331         '       '//temp_str(1)//' '//temp_str(2)//' '//temp_str(3)//new_line('a')
332     temp_str(1)=self%Z0
333     outext = outext//'       Z0 = '//temp_str(1)//' m'//new_line('a')
334     temp_str(1)=self%Rho_ref
335     outext = outext//'       Rho_ref = '//temp_str(1)//' kg/m^3'
336 
337     call log%put(outext,.false.)
```

6.10.2.4 printsimdefs()

```
subroutine simulation_globals_mod::printsimdefs (
            class(simdefs_t), intent(in) self )  [private]
```

Public simulation definitions printing routine.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 441 of file simulation_globals.f90.

```
441     implicit none
442     class(simdefs_t), intent(in) :: self
443     type(string) :: outext
444     type(string) :: temp_str(3)
445 
446     temp_str(1)=self%Dp
447     outext = '       Initial resolution is '//temp_str(1)//' m'//new_line('a')
448     temp_str(1)=self%dt
449     outext = '       Timestep is '//temp_str(1)//' s'//new_line('a')
450     temp_str(1)=self%Pointmin%x
451     temp_str(2)=self%Pointmin%y
452     temp_str(3)=self%Pointmin%z
453     outext = outext//'       Pointmin (BB) is '//new_line('a')//&
454         '       '//temp_str(1)//' '//temp_str(2)//' '//temp_str(3)//new_line('a')
455     temp_str(1)=self%Pointmax%x
456     temp_str(2)=self%Pointmax%y
457     temp_str(3)=self%Pointmax%z
458     outext = outext//'       Pointmax (BB) is '//new_line('a')//&
459         '       '//temp_str(1)//' '//temp_str(2)//' '//temp_str(3)//new_line('a')
460     temp_str(1)=self%blocksize%x
461     temp_str(2)=self%blocksize%y
462     outext = outext//'       Blocks are sized '//new_line('a')//&
463         '       '//temp_str(1)//' X '//temp_str(2)
464 
465     call log%put(outext,.false.)
```

6.10.2.5 printsimparameters()

```
subroutine simulation_globals_mod::printsimparameters (
             class(parameters_t), intent(inout) self )  [private]
```

Parameter printing method.

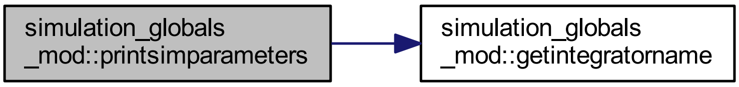
Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 207 of file simulation_globals.f90.

```
207     implicit none
208     class(parameters_t), intent(inout) :: self
209     type(string) :: outext
210     type(string) :: temp_str
211     call getintegratorname(temp_str,self%Integrator)
212     outext = '      Integrator scheme is '//temp_str//new_line('a')
213     temp_str=self%CFL
214     outext = outext//'       CFL = '//temp_str//new_line('a')
215     temp_str=self%WarmUpTime
216     outext = outext//'       WarmUpTime = '//temp_str//' s'//new_line('a')
217     temp_str=self%TimeMax
218     outext = outext//'       TimeMax = '//temp_str//' s'//new_line('a')
219     temp_str=self%TimeOut
220     outext = outext//'       TimeOut = '//temp_str//' Hz'
221     call log%put(outext,.false.)
```

Here is the call graph for this function:



6.10.2.6 setblocksize()

```
subroutine simulation_globals_mod::setblocksize (
             class(simdefs_t), intent(inout) self,
             type(vector) bsize )  [private]
```

blocksize box setting routine.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*bsize*	

Definition at line 424 of file simulation_globals.f90.

```
424     implicit none
425     class(simdefs_t), intent(inout) :: self
426     type(vector) :: bsize
427     integer :: sizem
428     self%blocksize = bsize
429     sizem = sizeof(bsize)
430     call simmemory%adddef(sizem)
```

6.10.2.7 setboundingbox()

```
subroutine simulation_globals_mod::setboundingbox (
            class(simdefs_t), intent(inout) self,
            type(string), intent(in) point_,
            type(vector) coords )  [private]
```

Bounding box setting routine.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*point_,coords*	

Definition at line 400 of file simulation_globals.f90.

```
400     implicit none
401     class(simdefs_t), intent(inout) :: self
402     type(string), intent(in) :: point_
403     type(vector) :: coords
404     integer :: sizem
405     if (point_%chars() == "pointmin") then
406         self%Pointmin= coords
407     elseif (point_%chars() == "pointmax") then
408         self%Pointmax= coords
409     endif
410     sizem=sizeof(coords)
411     call simmemory%adddef(sizem)
```

6.10.2.8 setdefaults()

```
subroutine simulation_globals_mod::setdefaults (
            class(globals_class), intent(inout) self )  [private]
```

Globals default setting routine.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 104 of file simulation_globals.f90.

```
104       implicit none
105       class(globals_class), intent(inout) :: self
106       integer :: sizem
107       !parameters
108       self%Parameters%Integrator = 1
109       self%Parameters%CFL = 0.5
110       self%Parameters%WarmUpTime = 0.0
111       self%Parameters%TimeOut = mv
112       self%Parameters%TimeOut = mv
113       !Simulation definitions
114       self%SimDefs%autoblocksize =.true.
115       self%SimDefs%blocksize = 0.0
116       self%SimDefs%numblocks = 16  !placeholder number, should be numThreads or numProcesses or computed by user dimensions
117       self%SimDefs%Dp = mv
118       self%SimDefs%dt = mv
119       self%SimDefs%Pointmin = 0.0
120       self%SimDefs%Pointmax = 0.0
121       !simulation constants
122       self%Constants%Gravity= 0.0*ex + 0.0*ey -9.81*ez
123       self%Constants%Z0 = 0.0
124       self%Constants%Rho_ref = 1000.0
125       !filenames
126       self%FileNames%mainxmlfilename = 'not_set'
127       self%FileNames%propsxmlfilename = 'not_set'
128       self%FileNames%tempfilename = 'not_set'
129       !global time
130       self%SimTime = 0.0
131 
132       sizem=sizeof(self)
133       call simmemory%adddef(sizem)
134 
```

6.10.2.9 setdp()

```
subroutine simulation_globals_mod::setdp (
              class(simdefs_t), intent(inout) self,
              type(string), intent(in) read_dp )  [private]
```

Dp setting routine.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*read_dp*	

Definition at line 350 of file simulation_globals.f90.

```
350     implicit none
351     class(simdefs_t), intent(inout) :: self
352     type(string), intent(in) :: read_dp
353     type(string) :: outext
354     integer :: sizem
```

```
355     self%Dp=read_dp%to_number(kind=1._r4p)
356     if (self%Dp.le.0.0) then
357         outext='Dp must be positive and non-zero, stopping'
358         call log%put(outext)
359         stop
360     endif
361     sizem = sizeof(self%Dp)
362     call simmemory%adddef(sizem)
```

6.10.2.10 setdt()

```
subroutine simulation_globals_mod::setdt (
            class(simdefs_t), intent(inout) self,
            type(string), intent(in) read_dt )  [private]
```

Dt setting routine.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*read_dt*	

Definition at line 375 of file simulation_globals.f90.

```
375     implicit none
376     class(simdefs_t), intent(inout) :: self
377     type(string), intent(in) :: read_dt
378     type(string) :: outext
379     integer :: sizem
380     self%dt=read_dt%to_number(kind=1._r4p)
381     if (self%dt.le.0.0) then
382         outext='dt must be positive and non-zero, stopping'
383         call log%put(outext)
384         stop
385     endif
386     sizem = sizeof(self%dt)
387     call simmemory%adddef(sizem)
```

6.10.2.11 setgravity()

```
subroutine simulation_globals_mod::setgravity (
            class(constants_t), intent(inout) self,
            type(vector), intent(in) grav )  [private]
```

Gravity setting routine.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*grav*	

Definition at line 255 of file simulation_globals.f90.

```
255     implicit none
256     class(constants_t), intent(inout) :: self
257     type(vector), intent(in) :: grav
258     integer :: sizem
259     type(string) :: outext
260     self%Gravity= grav
261     if (grav%x==mv) then !Gravity was not read, setting default
262         self%Gravity= -9.81*ez
263         outext = '       Gravity not specified, setting to default value = (0,0,-9.81)'
264         call log%put(outext,.false.)
265     endif
266     sizem=sizeof(self%Gravity)
267     call simmemory%adddef(sizem)
```

6.10.2.12 setparameter()

```
subroutine simulation_globals_mod::setparameter (
               class(parameters_t), intent(inout) self,
               type(string), intent(in) parmkey,
               type(string), intent(in) parmvalue )  [private]
```

Private parameter setting method. Builds the simulation parametric space from the input case file.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*parmkey,parmvalue*	

Definition at line 147 of file simulation_globals.f90.

```
147     implicit none
148     class(parameters_t), intent(inout) :: self
149     type(string), intent(in) :: parmkey
150     type(string), intent(in) :: parmvalue
151     character(80) :: value
152     integer :: sizem
153     !add new parameters to this search
154     if (parmkey%chars()=="Integrator") then
155         self%Integrator=parmvalue%to_number(kind=1_i1p)
156         sizem=sizeof(self%Integrator)
157     elseif(parmkey%chars()=="CFL") then
158         self%CFL=parmvalue%to_number(kind=1._r4p)
159         sizem=sizeof(self%CFL)
160     elseif(parmkey%chars()=="WarmUpTime") then
161         self%WarmUpTime=parmvalue%to_number(kind=1._r4p)
162         sizem=sizeof(self%WarmUpTime)
163     elseif(parmkey%chars()=="TimeMax") then
164         self%TimeMax=parmvalue%to_number(kind=1._r4p)
165         sizem=sizeof(self%TimeMax)
166     elseif(parmkey%chars()=="TimeOut") then
```

```
        self%TimeOut=parmvalue%to_number(kind=1._r4p)
        sizem=sizeof(self%TimeOut)
    endif
    call simmemory%adddef(sizem)

```

6.10.2.13 setrho()

```
subroutine simulation_globals_mod::setrho (
            class(constants_t), intent(inout) self,
            type(string), intent(in) read_rho )  [private]
```

Rho_Ref setting routine.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*read_rho*	

Definition at line 299 of file simulation_globals.f90.

```
299     implicit none
300     class(constants_t), intent(inout) :: self
301     type(string), intent(in) :: read_rho
302     type(string) :: outext
303     integer :: sizem
304     self%Rho_ref=read_rho%to_number(kind=1._r4p)
305     if (self%Rho_ref.le.0.0) then
306         outext='Rho_ref must be positive and non-zero, stopping'
307         call log%put(outext)
308         stop
309     endif
310     sizem = sizeof(self%Rho_ref)
311     call simmemory%adddef(sizem)
```

6.10.2.14 setz0()

```
subroutine simulation_globals_mod::setz0 (
            class(constants_t), intent(inout) self,
            type(string), intent(in) read_z0 )  [private]
```

Z0 setting routine.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*read_z0*	

Definition at line 280 of file simulation_globals.f90.

```
280     implicit none
281     class(constants_t), intent(inout) :: self
282     type(string), intent(in) :: read_z0
283     integer :: sizem
284     self%Z0=read_z0%to_number(kind=1._r4p)
285     sizem = sizeof(self%Z0)
286     call simmemory%adddef(sizem)
```

6.10.3 Variable Documentation

6.10.3.1 globals

```
type(globals_class), public simulation_globals_mod::globals
```

Definition at line 89 of file simulation_globals.f90.

```
89     type(globals_class) :: Globals
```

6.11 simulation_logger_mod Module Reference

Module to hold all the simulation logger related definitions and methods.

Data Types

- type logger_class

Functions/Subroutines

- subroutine initlog (self, outpath)

  Log file initizalization routine.
- subroutine closelog (self)

  Log file closure routine.
- subroutine put_inlog (self, tologstr, timeoption)

  Log serialization routine.
- subroutine, public gettimestamp (timestamp)

  Public timestamp builder.

Variables

- type(logger_class), public log

6.11.1 Detailed Description

Module to hold all the simulation logger related definitions and methods.

Author

Ricardo Birjukovs Canelas

6.11.2 Function/Subroutine Documentation

6.11.2.1 closelog()

```
subroutine simulation_logger_mod::closelog (
            class(logger_class), intent(inout) self )  [private]
```

Log file closure routine.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 77 of file simulation_logger.f90.

```
77     implicit none
78     class(logger_class), intent(inout) :: self
79     close(self%log_unit)
```

6.11.2.2 gettimestamp()

```
subroutine, public simulation_logger_mod::gettimestamp (
            type(string), intent(out) timestamp )
```

Public timestamp builder.

Author

Ricardo Birjukovs Canelas - MARETEC

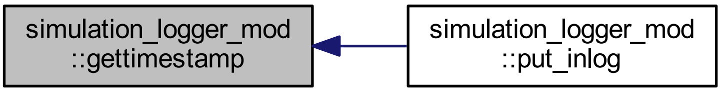
Parameters

in	*timestamp*	

Definition at line 120 of file simulation_logger.f90.

```
    implicit none
    type(string), intent(out) :: timestamp
    character(80) :: temp(8)
    integer :: values(8),i

    call date_and_time(values=values)
    do i=1,8
        write(temp(i),*) values(i)
    enddo
    timestamp=trim(adjustl(temp(1)))//'-'//trim(adjustl(temp(2)))//'-'//trim(adjustl(temp(3)))//' @'//trim(adjustl(temp(5)))//':'//trim(adjustl(temp(6)))//':'//trim(adjustl(temp(7)))
```

Here is the caller graph for this function:


simulation_logger_mod ::gettimestamp
simulation_logger_mod ::put_inlog

6.11.2.3 initlog()

```
subroutine simulation_logger_mod::initlog (
            class(logger_class), intent(inout) self,
            type(string), intent(in) outpath )  [private]
```

Log file initizalization routine.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*outpath*	
in	*outpath*	output path were to point the logger

Definition at line 58 of file simulation_logger.f90.

```
    implicit none
    class(logger_class), intent(inout) :: self
    type(string), intent(in) :: outpath
    type(string) :: logfile

    logfile = outpath//'MOHIDLagrangianRun.out'
    self%log_unit = 0
    open (unit=self%log_unit,file=logfile%chars(),action="write",status="replace")

```

6.11.2.4 put_inlog()

```
subroutine simulation_logger_mod::put_inlog (
            class(logger_class), intent(in) self,
            type(string), intent(inout) tologstr,
            logical, intent(in), optional timeoption )  [private]
```

Log serialization routine.

Author

Ricardo Birjukovs Canelas - MARETEC

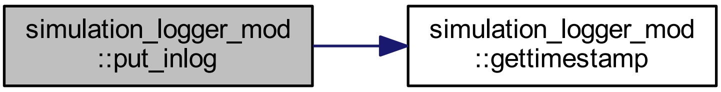
Parameters

in	*tologstr,timeoption*	

Definition at line 92 of file simulation_logger.f90.

```
92      implicit none
93      class(logger_class), intent(in) :: self
94      type(string), intent(inout) :: tologstr
95      logical, intent(in), optional :: timeoption
96      type(string) :: timestamp
97
98      call gettimestamp(timestamp)
99      if (present(timeoption)) then
100        if (.not.timeoption) then
101          timestamp=''
102        endif
103      endif
104      tologstr=timestamp//' '//tologstr
105      write(self%log_unit,"(A)") tologstr%chars()
106      print'(A)', tologstr%chars()
107
```

Here is the call graph for this function:


simulation_logger_mod ::put_inlog
simulation_logger_mod ::gettimestamp

6.11.3 Variable Documentation

6.11.3.1 log

```
type(logger_class), public simulation_logger_mod::log
```

Definition at line 38 of file simulation_logger.f90.

```
38      type(logger_class) :: Log
```

6.12 simulation_memory_mod Module Reference

Module to hold the simulation memory managment class and its methods.

Data Types

- type memory_t

Functions/Subroutines

- subroutine initializememory (self)

  Private memory logger initialization method.
- subroutine getotal (self, size)

  Private method to retreive the total size of the allocated memory.
- subroutine addblock (self, size)

  Private method to add the size of a Block to the memory log.
- subroutine addsource (self, size)

  Private method to add the size of a Source to the memory log.
- subroutine addtracer (self, size)

  Private method to add the size of a Tracer to the memory log.
- subroutine removetracer (self, size)

  Private method to remove the size of a Tracer from the memory log.
- subroutine adddef (self, size)

  Private method to add the size of a definition to the memory log.
- subroutine printmemory (self)

  Method to print the total allocated memory.
- subroutine printmemorydetailed (self)

  Private method to print the allocated memory.

Variables

- type(memory_t), public simmemory

6.12.1 Detailed Description

Module to hold the simulation memory managment class and its methods.

Author

Ricardo Birjukovs Canelas

6.12.2 Function/Subroutine Documentation

6.12.2.1 addblock()

```
subroutine simulation_memory_mod::addblock (
             class(memory_t), intent(inout) self,
             integer, intent(in) size )  [private]
```

Private method to add the size of a Block to the memory log.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 91 of file simulation_memory.f90.

```
91      implicit none
92      class(memory_t), intent(inout) :: self
93      integer, intent(in) :: size
94      self%size_of_blocks = self%size_of_blocks + size
```

6.12.2.2 adddef()

```
subroutine simulation_memory_mod::adddef (
             class(memory_t), intent(inout) self,
             integer, intent(in) size )  [private]
```

Private method to add the size of a definition to the memory log.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 147 of file simulation_memory.f90.

```
147     implicit none
148     class(memory_t), intent(inout) :: self
149     integer, intent(in) :: size
150     self%size_of_defs = self%size_of_defs + size
```

6.12.2.3 addsource()

```
subroutine simulation_memory_mod::addsource (
             class(memory_t), intent(inout) self,
             integer, intent(in) size )  [private]
```

Private method to add the size of a Source to the memory log.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 105 of file simulation_memory.f90.

```
105     implicit none
106     class(memory_t), intent(inout) :: self
107     integer, intent(in) :: size
108     self%size_of_sources = self%size_of_sources + size
```

6.12.2.4 addtracer()

```
subroutine simulation_memory_mod::addtracer (
            class(memory_t), intent(inout) self,
            integer, intent(in) size )  [private]
```

Private method to add the size of a Tracer to the memory log.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 119 of file simulation_memory.f90.

```
119     implicit none
120     class(memory_t), intent(inout) :: self
121     integer, intent(in) :: size
122     self%size_of_tracers = self%size_of_tracers + size
```

6.12.2.5 getotal()

```
subroutine simulation_memory_mod::getotal (
            class(memory_t), intent(inout) self,
            integer, intent(out) size )  [private]
```

Private method to retreive the total size of the allocated memory.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 77 of file simulation_memory.f90.

```
77     implicit none
78     class(memory_t), intent(inout) :: self
79     integer, intent(out) :: size
80     size = self%size_of_sources + self%size_of_tracers + self%size_of_defs + self%size_of_blocks
```

6.12.2.6 initializememory()

```
subroutine simulation_memory_mod::initializememory (
            class(memory_t), intent(inout) self )  [private]
```

Private memory logger initialization method.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 61 of file simulation_memory.f90.

```
61     implicit none
62     class(memory_t), intent(inout) :: self
63     self%size_of_sources = 0
64     self%size_of_tracers = 0
65     self%size_of_defs = 0
66     self%size_of_blocks = 0
```

6.12.2.7 printmemory()

```
subroutine simulation_memory_mod::printmemory (
            class(memory_t), intent(inout) self )  [private]
```

Method to print the total allocated memory.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 161 of file simulation_memory.f90.

```
161     implicit none
162     class(memory_t), intent(inout) :: self
163     integer :: size
164     real(prec) :: sizemb
165     type(string) :: outext,temp
166     call self%getotal(size)
167     sizemb = size*1e-6
168     temp= sizemb
169     outext='->Total allocated memory: '//temp//' mb'
170     call log%put(outext)
```

6.12.2.8 printmemorydetailed()

```
subroutine simulation_memory_mod::printmemorydetailed (
            class(memory_t), intent(inout) self )  [private]
```

Private method to print the allocated memory.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 181 of file simulation_memory.f90.

```
181     implicit none
182     class(memory_t), intent(inout) :: self
183     integer :: size
184     real(prec) :: sizemb
185     type(string) :: outext,temp(5)
186
187     call self%getotal(size)
188     sizemb = size*1e-6
189     temp(1)= sizemb
190     sizemb = self%size_of_sources*1e-6
191     temp(2)= sizemb
192     sizemb = self%size_of_tracers*1e-6
193     temp(3)= sizemb
194     sizemb = self%size_of_defs*1e-6
195     temp(4)= sizemb
196     sizemb = self%size_of_blocks*1e-6
197     temp(5)= sizemb
198
199     outext='->Total allocated memory: '//temp(1)//' mb'//new_line('a')//&
200         '       Allocated memory for Blocks  = '//temp(5)//' mb'//new_line('a')//&
201         '       Allocated memory for Sources = '//temp(2)//' mb'//new_line('a')//&
202         '       Allocated memory for Tracers = '//temp(3)//' mb'//new_line('a')//&
203         '       Allocated memory for Consts  = '//temp(4)//' mb'
204     call log%put(outext)
205
```

6.12.2.9 removetracer()

```
subroutine simulation_memory_mod::removetracer (
            class(memory_t), intent(inout) self,
            integer, intent(in) size ) [private]
```

Private method to remove the size of a Tracer from the memory log.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 133 of file simulation_memory.f90.

```
133     implicit none
134     class(memory_t), intent(inout) :: self
135     integer, intent(in) :: size
136     self%size_of_tracers = self%size_of_tracers - size
```

6.12.3 Variable Documentation

6.12.3.1 simmemory

```
type(memory_t), public simulation_memory_mod::simmemory
```

Definition at line 46 of file simulation_memory.f90.

```
46     type(memory_t) :: SimMemory
```

6.13 simulation_mod Module Reference

Module to hold the simulation class and its methods.

Data Types

- type simulation_class

Functions/Subroutines

- subroutine run (self)

    Simulation run method. Runs the initialized case main time cycle.
- subroutine initsimulation (self, casefilename, outpath)

    Simulation initialization method. Effectively builds and populates the simulation objects that will be used latter on.
- subroutine distributesources (self)

    Simulation to distribute the Sources to the blocks.
- subroutine decomposedomain (self)

    Simulation method to do domain decomposition and define the Blocks.
- subroutine closesimulation (self)

    Simulation finishing method. Closes output files and writes the final messages.

6.13.1 Detailed Description

Module to hold the simulation class and its methods.

Author

Ricardo Birjukovs Canelas

6.13.2 Function/Subroutine Documentation

6.13.2.1 closesimulation()

```
subroutine simulation_mod::closesimulation (
            class(simulation_class), intent(inout) self )  [private]
```

Simulation finishing method. Closes output files and writes the final messages.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 199 of file simulation.f90.

```
199     implicit none
200     class(simulation_class), intent(inout) :: self
201     type(string) :: outext
202
203     outext='Simulation ended, freeing resources. See you next time'
204     call log%put(outext)
205     call log%finalize()
206
```

6.13.2.2 decomposedomain()

```
subroutine simulation_mod::decomposedomain (
            class(simulation_class), intent(inout) self )  [private]
```

Simulation method to do domain decomposition and define the Blocks.

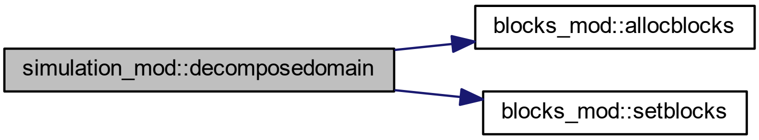
Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 175 of file simulation.f90.

```
implicit none
class(simulation_class), intent(inout) :: self
type(string) :: outext

if (globals%SimDefs%autoblocksize) then
    call allocblocks(globals%SimDefs%numblocks)
else
    outext='[DecomposeDomain]: Only automatic Block sizing at the moment, stoping'
    call log%put(outext)
    stop
end if
! Initializing the blocks
call setblocks(globals%SimDefs%autoblocksize,globals%SimDefs%numblocks,self%nbx,self%nby)

```

Here is the call graph for this function:



6.13.2.3 distributesources()

```
subroutine simulation_mod::distributesources (
            class(simulation_class), intent(inout) self )  [private]
```

Simulation to distribute the Sources to the blocks.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 131 of file simulation.f90.

```
implicit none
class(simulation_class), intent(inout) :: self
type(string) :: outext
integer :: i, ix, iy, blk
real(prec) :: dx, dy

!this is easy because all the blocks are the same
dx = dblock(1)%extents%size%x
dy = dblock(1)%extents%size%y
!iterate every Source to distribute
do i=1, size(tempsources%src)
    !finding the 2D coordinates of the corresponding Block
```

```
        ix = min(int((tempsources%src(i)%now%pos%x + bbox%offset%x)/dx) + 1, self%nbx)
        iy = min(int((tempsources%src(i)%now%pos%y + bbox%offset%y)/dy) + 1, self%nby)
        print*, 'Source position'
        print*, tempsources%src(i)%now%pos
        print*, 'Source grid position'
        print*, ix, iy
        !Converting to the 1D index - Notice how the blocks were built in [Blocks::setBlocks]
        blk = 2*ix + iy -2
        print*, blk
        if (blk > size(dblock)) then
            outext='[DistributeSources]: problem in getting correct Block index, stoping'
            call log%put(outext)
            stop
        end if
        call dblock(blk)%putSource(tempsources%src(i))
    end do
    call tempsources%finalize() !destroying the temporary Sources now they are shipped to the Blocks
    do i=1, size(dblock)
        call dblock(i)%detailedprint()
    enddo

```

6.13.2.4 initsimulation()

subroutine simulation_mod::initsimulation (
 class(simulation_class), intent(inout) *self,*
 type(string), intent(in) *casefilename,*
 type(string), intent(in) *outpath*) [private]

Simulation initialization method. Effectively builds and populates the simulation objects that will be used latter on.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

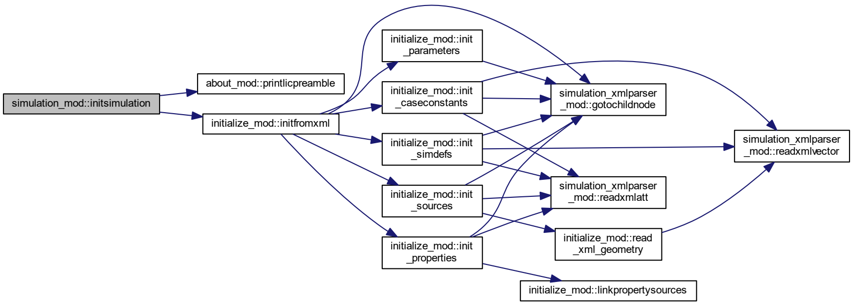
in	*casefilename,outpath*	
in	*casefilename*	case file name
in	*outpath*	Output path

Definition at line 81 of file simulation.f90.

```
81      implicit none
82      class(simulation_class), intent(inout) :: self
83      type(string), intent(in) :: casefilename
84      type(string), intent(in) :: outpath
85      type(string) :: outext
86
87      ! Initialize logger
88      call log%initialize(outpath)
89      !Print licences and build info
90      call printlicpreamble
91
92      !setting every global variable and input parameter to their default
93      call globals%initialize()
94      !initializing memory log
95      call simmemory%initialize()
96      !initializing geometry class
97      call geometry%initialize()
98
99      !Check if case file has .xml extension
100      if (casefilename%extension() == '.xml') then
101          ! Initialization routines to build the simulation from the input case file
```

```
        call initfromxml(casefilename)
    else
        outext='[initSimulation]: only .xml input files are supported at the time. Stopping'
        call log%put(outext)
        stop
    endif
    !Case was read and now we can build/initialize our simulation objects that are case-dependent

    !initilize simulation bounding box
    call bbox%initialize()
    !call BBox%print()

    call self%decompose()

    call self%DistributeSources()

    !printing memory occupation at the time
    call simmemory%detailedprint()

```

Here is the call graph for this function:


simulation_mod::initsimulation
about_mod::printlicpreamble
initialize_mod::initfromxml
initialize_mod::init _parameters
initialize_mod::init _caseconstants
initialize_mod::init _simdefs
initialize_mod::init _sources
initialize_mod::init _properties
simulation_xmlparser _mod::gotochildnode
simulation_xmlparser _mod::readxmlatt
initialize_mod::read _xml_geometry
initialize_mod::linkpropertysources
simulation_xmlparser _mod::readxmlvector

6.13.2.5 run()

subroutine simulation_mod::run (
class(simulation_class), intent(inout) *self*) [private]

Simulation run method. Runs the initialized case main time cycle.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 56 of file simulation.f90.

```
    implicit none
    class(simulation_class), intent(inout) :: self
    type(string) :: outext

    !main time cycle
    do while (globals%SimTime .LT. globals%Parameters%TimeMax)

        !Do your Lagrangian things here :D

        globals%SimTime = globals%SimTime + globals%SimDefs%dt
    enddo

```

6.14 simulation_precision_mod Module Reference

Module to control the precision of the variables trough the project.

Variables

- integer, parameter sp = kind(1._R4P)

  Simple precision definition switch.
- integer, parameter dp = kind(1._R8P)

  Double precision definition switch.
- integer, parameter, public prec = sp
- integer, parameter, public prec_time = sp
- integer, parameter, public prec_wrt = sp
- real(prec), parameter, public missing_value_default = -9999.0_dp
- real(prec), parameter, public mv = MISSING_VALUE_DEFAULT
- real(prec), parameter, public mv_int = int(MISSING_VALUE_DEFAULT)
- real(prec), parameter, public err_dist = 1E8_dp
- integer, parameter, public err_ind = -1
- integer, parameter, public char_len = 99

6.14.1 Detailed Description

Module to control the precision of the variables trough the project.

Author

Ricardo Birjukovs Canelas

6.14.2 Variable Documentation

6.14.2.1 char_len

```
integer, parameter, public simulation_precision_mod::char_len = 99
```

Definition at line 48 of file simulation_precision.f90.

```
48     integer, parameter :: CHAR_LEN = 99
```

6.14.2.2 dp

```
integer, parameter simulation_precision_mod::dp = kind(1._R8P)  [private]
```

Double precision definition switch.

Definition at line 31 of file simulation_precision.f90.

```
31     integer,  parameter :: dp  = kind(1._r8p)
```

6.14.2.3 err_dist

```
real(prec), parameter, public simulation_precision_mod::err_dist = 1E8_dp
```

Definition at line 44 of file simulation_precision.f90.

```
44     real(prec), parameter :: ERR_DIST = 1e8_dp
```

6.14.2.4 err_ind

```
integer, parameter, public simulation_precision_mod::err_ind = -1
```

Definition at line 45 of file simulation_precision.f90.

```
45     integer,  parameter   :: ERR_IND  = -1
```

6.14.2.5 missing_value_default

```
real(prec), parameter, public simulation_precision_mod::missing_value_default = -9999.0_dp
```

Definition at line 39 of file simulation_precision.f90.

```
39     real(prec), parameter :: MISSING_VALUE_DEFAULT = -9999.0_dp
```

6.14.2.6 mv

```
real(prec), parameter, public simulation_precision_mod::mv = MISSING_VALUE_DEFAULT
```

Definition at line 40 of file simulation_precision.f90.

```
40      real(prec), parameter :: MV     = missing_value_default
```

6.14.2.7 mv_int

```
real(prec), parameter, public simulation_precision_mod::mv_int = int(MISSING_VALUE_DEFAULT)
```

Definition at line 41 of file simulation_precision.f90.

```
41      real(prec), parameter :: MV_INT = int(missing_value_default)
```

6.14.2.8 prec

```
integer, parameter, public simulation_precision_mod::prec = sp
```

Definition at line 34 of file simulation_precision.f90.

```
34      integer,  parameter :: prec      = sp
```

6.14.2.9 prec_time

```
integer, parameter, public simulation_precision_mod::prec_time = sp
```

Definition at line 35 of file simulation_precision.f90.

```
35      integer,  parameter :: prec_time = sp
```

6.14.2.10 prec_wrt

```
integer, parameter, public simulation_precision_mod::prec_wrt = sp
```

Definition at line 36 of file simulation_precision.f90.

```
36      integer,  parameter :: prec_wrt  = sp
```

6.14.2.11 sp

```
integer, parameter simulation_precision_mod::sp = kind(1._R4P)  [private]
```

Simple precision definition switch.

Definition at line 30 of file simulation_precision.f90.

```
30     integer,  parameter :: sp  = kind(1._r4p)
```

6.15 simulation_xmlparser_mod Module Reference

Module with the simulation xml parsing related definitions and routines.

Functions/Subroutines

- subroutine, public readxmlatt (xmlnode, tag, att_name, att_value, read_flag, mandatory)

  Private attribute xml parser routine. In the format <Tag att_name="att_value".
- subroutine, public readxmlvector (xmlnode, tag, vec, read_flag, mandatory)

  Private vector xml parser routine. Vector must be in format <Tag x="vec%x" y="vec%y" z="vec%z">
- subroutine, public gotochildnode (currentNode, targetNode, targetNodeName, read_flag, mandatory)

  Private routine to retrieve a node within a node. Returns a nullifyed pointer if not found, stops if mandatory.

6.15.1 Detailed Description

Module with the simulation xml parsing related definitions and routines.

Author

Ricardo Birjukovs Canelas

6.15.2 Function/Subroutine Documentation

6.15.2.1 gotochildnode()

```
subroutine, public simulation_xmlparser_mod::gotochildnode (
             type(node), intent(in), pointer currentNode,
             type(node), intent(out), pointer targetNode,
             type(string), intent(in) targetNodeName,
             logical, intent(out), optional read_flag,
             logical, intent(in), optional mandatory )
```

Private routine to retrieve a node within a node. Returns a nullifyed pointer if not found, stops if mandatory.

Author

Ricardo Birjukovs Canelas - MARETEC

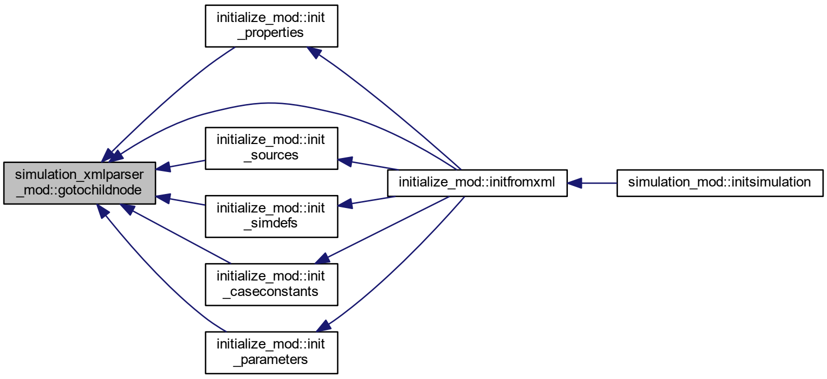
Parameters

in	*currentNode,targetNode,targetNodeName,mandatory*	
out	*read_flag*	Optional flag to capture read/non-read status
in	*mandatory*	Swich for optional or mandatory tags

Definition at line 162 of file simulation_xmlparser.f90.

```
162     implicit none
163     type(Node), intent(in), pointer :: currentNode
164     type(Node), intent(out), pointer :: targetNode
165     type(string), intent(in) :: targetNodeName
166     logical, intent(out), optional :: read_flag
167     logical, intent(in), optional :: mandatory
168
169     type(NodeList), pointer :: target_node_list
170     type(string) :: outext, nodename
171     integer :: i
172     logical :: target_node_exists
173
174     target_node_exists = .false.
175     target_node_list => getchildnodes(currentnode)
176     do i=0, getlength(target_node_list)-1
177         targetnode => item(target_node_list,i) !grabing a node
178         nodename = getlocalname(targetnode)  !finding its name
179         if (nodename == targetnodename) then !found our target node
180             target_node_exists = .true.
181             if (present(read_flag)) then
182               read_flag =.true.
183             endif
184             exit
185         endif
186     enddo
187     if (target_node_exists .eqv. .false.) then
188         nullify(targetnode)
189         if(present(mandatory)) then
190           if (mandatory.eqv..false.) then
191             outext='Could not find any node called "'//targetnodename//'" in the xml file, ignoring'
192             call log%put(outext)
193             if (present(read_flag)) then
194               read_flag =.false.
195             endif
196           else
197             outext='Could not find any node called "'//targetnodename//'" in the xml file, stoping'
198             call log%put(outext)
199             stop
200           endif
201         else
202           outext='Could not find any node called "'//targetnodename//'" in the xml file, stoping'
203           call log%put(outext)
204           stop
205         endif
206     endif
207
```

Here is the caller graph for this function:


initialize_mod::init_properties
simulation_xmlparser_mod::gotochildnode
initialize_mod::init_sources
initialize_mod::initfromxml
simulation_mod::initsimulation
initialize_mod::init_simdefs
initialize_mod::init_caseconstants
initialize_mod::init_parameters

6.15.2.2 readxmlatt()

```
subroutine, public simulation_xmlparser_mod::readxmlatt (
            type(node), intent(in), pointer xmlnode,
            type(string), intent(in) tag,
            type(string), intent(in) att_name,
            type(string), intent(out) att_value,
            logical, intent(out), optional read_flag,
            logical, intent(in), optional mandatory )
```

Private attribute xml parser routine. In the format <Tag att_name="att_value".

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

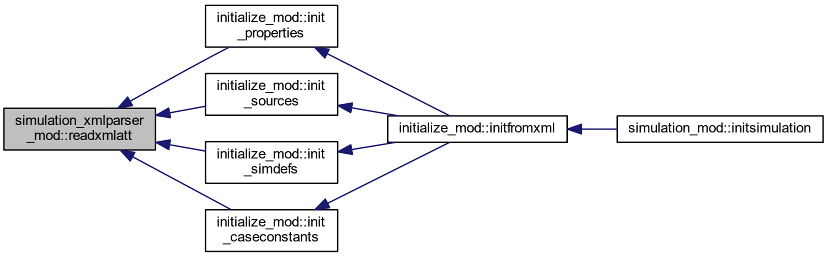
in	*xmlnode,tag,vec,mandatory*	
in	*xmlnode*	Working xml node
in	*tag*	Tag to search in xml node
in	*att_name*	Atribute name to collect from tag
out	*att_value*	Attribute value
out	*read_flag*	Optional flag to capture read/non-read status
in	*mandatory*	Swich for optional or mandatory tags

Definition at line 43 of file simulation_xmlparser.f90.

```
43      implicit none
44      type(Node), intent(in), pointer :: xmlnode
45      type(string), intent(in) :: tag
46      type(string), intent(in) :: att_name
47      type(string), intent(out) :: att_value
48      logical, intent(out), optional :: read_flag
49      logical, intent(in), optional :: mandatory
50
51      type(string) :: outext, nodename
52      character(80) :: att_value_chars
53      type(NodeList), pointer :: target_node_list, nodeChildren
54      type(Node), pointer :: nodedetail
55      logical :: validtag
56      integer :: i
57
58      validtag = .false.
59      nodechildren => getchildnodes(xmlnode) !getting all of the nodes bellow the main source node (all of
        it's private info)
60      do i=0, getlength(nodechildren)-1
61          nodedetail => item(nodechildren,i) !grabing a node
62          nodename = getlocalname(nodedetail)  !finding its name
63          if (nodename == tag) then
64              validtag=.true.
65              exit
66          endif
67      enddo
68      if (validtag) then
69          target_node_list => getelementsbytagname(xmlnode, tag%chars())   !searching for tags with the given
        name
70          nodedetail => item(target_node_list, 0)
71          call extractdataattribute(nodedetail, att_name%chars(), att_value_chars)
72          att_value=trim(att_value_chars)
73          if (present(read_flag)) then
```

```
            read_flag =.true.
          endif
      else
          if(present(mandatory)) then
              if(mandatory.eqv..false.) then
                if (present(read_flag)) then
                  read_flag =.false.
                endif
              endif
          else
              outext='Could not find any "'//tag//'" tag for xml node "'//getnodename(xmlnode)//'", stoping'
              call log%put(outext)
              stop
          endif
      endif

```

Here is the caller graph for this function:


initialize_mod::init_properties
initialize_mod::init_sources
simulation_xmlparser_mod::readxmlatt
initialize_mod::init_simdefs
initialize_mod::init_caseconstants
initialize_mod::initfromxml
simulation_mod::initsimulation

6.15.2.3 readxmlvector()

```
subroutine, public simulation_xmlparser_mod::readxmlvector (
            type(node), intent(in), pointer xmlnode,
            type(string), intent(in) tag,
            type(vector), intent(out) vec,
            logical, intent(out), optional read_flag,
            logical, intent(in), optional mandatory )
```

Private vector xml parser routine. Vector must be in format <Tag x="vec%x" y="vec%y" z="vec%z">

Author

Ricardo Birjukovs Canelas - MARETEC

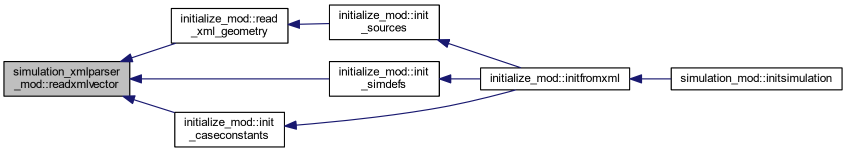
Parameters

in	*xmlnode,tag,vec,mandatory*	
in	*xmlnode*	Working xml node
in	*tag*	Tag to search in xml node
out	*vec*	Vector to fill with read contents
out	*read_flag*	Optional flag to capture read/non-read status
in	*mandatory*	Swich for optional or mandatory tags

Definition at line 102 of file simulation_xmlparser.f90.

```
102     implicit none
103     type(Node), intent(in), pointer :: xmlnode
104     type(string), intent(in) :: tag
105     type(vector), intent(out) :: vec
106     logical, intent(out), optional :: read_flag
107     logical, intent(in), optional :: mandatory
108
109     type(string) :: outext, nodename
110     type(NodeList), pointer :: target_node_list, nodeChildren
111     type(Node), pointer :: nodedetail
112     logical :: validtag
113     integer :: i
114
115     vec%x=mv !marking the array as not read
116     validtag = .false.
117     nodechildren => getchildnodes(xmlnode) !getting all of the nodes bellow the main source node (all of it's private info)
118     do i=0, getlength(nodechildren)-1
119         nodedetail => item(nodechildren,i) !grabing a node
120         nodename = getlocalname(nodedetail)  !finding its name
121         if (nodename == tag) then
122             validtag =.true.
123             exit
124         endif
125     enddo
126     if (validtag) then
127         target_node_list => getelementsbytagname(xmlnode, tag%chars())   !searching for tags with the given name
128         nodedetail => item(target_node_list, 0)
129         call extractdataattribute(nodedetail, "x", vec%x)
130         call extractdataattribute(nodedetail, "y", vec%y)
131         call extractdataattribute(nodedetail, "z", vec%z)
132         if (present(read_flag)) then
133           read_flag =.true.
134         endif
135     else
136         if(present(mandatory)) then
137             if(mandatory.eqv..false.) then
138               if (present(read_flag)) then
139                 read_flag =.false.
140               endif
141             endif
142         else
143             outext='Could not find any "'//tag//'" tag for xml node "'//getnodename(xmlnode)//'", stoping'
144             call log%put(outext)
145             stop
146         endif
147     endif
```

Here is the caller graph for this function:


initialize_mod::read_xml_geometry
initialize_mod::init_sources
simulation_xmlparser_mod::readxmlvector
initialize_mod::init_simdefs
initialize_mod::initfromxml
simulation_mod::initsimulation
initialize_mod::init_caseconstants

6.16 sources_array_mod Module Reference

Data Types

- type sourcearray

Functions/Subroutines

- subroutine print_sourcearray (this)
- subroutine print_sourcearray_element (this, index)

6.16.1 Function/Subroutine Documentation

6.16.1.1 print_sourcearray()

```
subroutine sources_array_mod::print_sourcearray (
            class(sourcearray), intent(in) this )  [private]
```

Definition at line 36 of file sources_array.f90.

```
36     class(SourceArray), intent(in) :: this
37     class(*), pointer :: curr
38     integer :: i
39     do i=1, this%usedLength
40         curr => this%get(i)
41         select type(curr)
42         type is (source_class)
43             call curr%print()
44             class default
45             stop '[print_SourceArray]: unexepected type of content: not a Source or derived type'
46         end select
47     end do
```

6.16.1.2 print_sourcearray_element()

```
subroutine sources_array_mod::print_sourcearray_element (
            class(sourcearray), intent(in) this,
            integer, intent(in) index )  [private]
```

Definition at line 51 of file sources_array.f90.

```
51     class(SourceArray), intent(in) :: this
52     integer, intent(in) :: index
53     class(*), pointer :: curr
54     if (index .le. this%usedLength) then
55         curr => this%get(index)
56         select type(curr)
57         type is (source_class)
58             call curr%print()
59             class default
60             stop '[print_SourceArray_Element]: unexepected type of content, not a Source or derived type'
61         end select
62     else
63         stop '[print_SourceArray_Element]: index out of bounds'
64     endif
```

6.17 sources_mod Module Reference

Module that defines a source class and related methods.

Data Types

- type source_class

  Type - The source class.
- type source_group_class
- type source_par
- type source_state

  Type - state variables of a source object.
- type source_stats

  Type - statistical variables of a source object.
- type source_stencil

  Type - holder for the tracer creation stencil of the source.

Functions/Subroutines

- subroutine initsources (self, nsources)

  source allocation routine - allocates sources objects
- subroutine killsources (self)

  source group destructor - deallocates sources objects
- subroutine setprops (self, srcid_str, ptype, pname)

  source property setting routine, calls source by id to set its properties
- subroutine initializesource (src, id, name, emitting_rate, start, finish, source_geometry, shapetype)

  source inititialization proceadure - initializes Source variables
- subroutine linkproperty (src, ptype, pname)

  source property setting proceadure - initializes Source variables
- subroutine printsource (src)

  source print routine - prints a source info on console/log

Variables

- type(source_group_class), public tempsources

6.17.1 Detailed Description

Module that defines a source class and related methods.

Author

Ricardo Birjukovs Canelas

6.17.2 Function/Subroutine Documentation

6.17.2.1 initializesource()

```
subroutine sources_mod::initializesource (
            class(source_class) src,
            integer, intent(in) id,
            type(string), intent(in) name,
            real(prec), intent(in) emitting_rate,
            real(prec), intent(in) start,
            real(prec), intent(in) finish,
            type(string), intent(in) source_geometry,
            class(shape), intent(in) shapetype )  [private]
```

source inititialization proceadure - initializes Source variables

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*src,id,name,emitting_rate,source_geometry*	

Definition at line 191 of file sources.f90.

```
    implicit none
    class(source_class) :: src
    integer, intent(in) :: id
    type(string), intent(in) :: name
    real(prec), intent(in) :: emitting_rate
    real(prec), intent(in) :: start
    real(prec), intent(in) :: finish
    type(string), intent(in) :: source_geometry
    class(shape), intent(in) :: shapetype

    integer :: sizem, i
    type(string) :: outext
    integer :: err

    !Setting parameters
    src%par%id=id
    src%par%emitting_rate=emitting_rate
    src%par%startime=start
    src%par%stoptime=finish
    src%par%name=name
    src%par%source_geometry=source_geometry
    allocate(src%par%geometry, source=shapetype)
    src%par%property_type = "pure" ! pure Lagrangian trackers by default
    src%par%property_name = "pure"
    !Setting state variables
    src%now%age=0.0
    src%now%active=.false. !disabled by default
    src%now%pos=src%par%geometry%pt !coords of the Source (meaning depends on the geometry type!)
    !setting statistical samplers
    src%stats%particles_emitted=0
    src%stats%acc_T=0.0
    src%stats%ns=0
    !setting stencil variables
    src%stencil%np = geometry%fillsize(src%par%geometry)
    allocate(src%stencil%ptlist(src%stencil%np), stat=err)
    if(err/=0)then
        outext='Cannot allocate point list for Source '// src%par%name //', stoping'
        call log%put(outext)
        stop
    endif
    call geometry%fill(src%par%geometry, src%stencil%np, src%stencil%ptlist)

    sizem = sizeof(src)
    call simmemory%addsource(sizem)
    call src%print()

```

```
!DBG
!do i=1, src%stencil%np
!print*, src%stencil%ptlist(i)
!end do
```

6.17.2.2 initsources()

```
subroutine sources_mod::initsources (
            class(source_group_class), intent(inout) self,
            integer, intent(in) nsources )  [private]
```

source allocation routine - allocates sources objects

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*nsources*	

Definition at line 101 of file sources.f90.

```
implicit none
class(source_group_class), intent(inout) :: self
integer, intent(in) :: nsources
integer err
type(string) :: outext, temp
allocate(self%src(nsources), stat=err)
if(err/=0)then
    outext='[initSources]: Cannot allocate Sources, stoping'
    call log%put(outext)
    stop
else
    temp = nsources
    outext = 'Allocated '// temp // ' Sources.'
    call log%put(outext)
endif
```

6.17.2.3 killsources()

```
subroutine sources_mod::killsources (
            class(source_group_class), intent(inout) self )  [private]
```

source group destructor - deallocates sources objects

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 127 of file sources.f90.

```
127     implicit none
128     class(source_group_class), intent(inout) :: self
129     integer err
130     type(string) :: outext
131     if (ALLOCATED(self%src)) deallocate(self%src, stat=err)
132     if(err/=0)then
133         outext='[killSources]: Cannot deallocate Sources, stoping'
134         call log%put(outext)
135         stop
136     endif
```

6.17.2.4 linkproperty()

```
subroutine sources_mod::linkproperty (
            class(source_class) src,
            type(string), intent(in) ptype,
            type(string), intent(in) pname )   [private]
```

source property setting proceadure - initializes Source variables

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*src,ptype,pname*	

Definition at line 254 of file sources.f90.

```
254     implicit none
255     class(source_class) :: src
256     type(string), intent(in) :: ptype
257     type(string), intent(in) :: pname
258     src%par%property_type = ptype
259     src%par%property_name = pname
```

6.17.2.5 printsource()

```
subroutine sources_mod::printsource (
            class(source_class) src )   [private]
```

source print routine - prints a source info on console/log

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*src*	

Definition at line 272 of file sources.f90.

```
    implicit none
    class(source_class) :: src

    type(string) :: outext
    type(string) :: temp_str(3)

    temp_str(1)=src%par%id
    outext = '-->Source '//src%par%name//' allocated'//new_line('a')//&
        '       Id = '//temp_str(1)//new_line('a')//&
        '       Geometry type is '//src%par%source_geometry//new_line('a')
    temp_str(1)=src%now%pos%x
    temp_str(2)=src%now%pos%y
    temp_str(3)=src%now%pos%z
    outext = outext//'       Initially at coordinates'//new_line('a')//&
        '       '//temp_str(1)//' '//temp_str(2)//' '//temp_str(3)//new_line('a')
    temp_str(1)=src%par%emitting_rate
    temp_str(2)=src%stencil%np
    outext = outext//'       Emitting '//temp_str(2)//' tracers at a rate of '//temp_str(1)//' Hz'//new_line('a')
    temp_str(1)=src%par%startime
    temp_str(2)=src%par%stoptime
    outext = outext//'       Active from '//temp_str(1)//' to '//temp_str(2)//' seconds'

    call log%put(outext,.false.)

```

6.17.2.6 setprops()

```
subroutine sources_mod::setprops (
            class(source_group_class), intent(inout) self,
            type(string), intent(in) srcid_str,
            type(string), intent(in) ptype,
            type(string), intent(in) pname )   [private]
```

source property setting routine, calls source by id to set its properties

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*srcid,ptype,pname*	
in	*srcid_str*	Source id tag
in	*ptype*	Property type to set
in	*pname*	Property name to set

Definition at line 150 of file sources.f90.

```
    implicit none
    class(source_group_class), intent(inout) :: self
    type(string), intent(in) :: srcid_str
    type(string), intent(in) :: ptype
    type(string), intent(in) :: pname

    integer :: srcid
    type(string) :: outext, temp
    integer :: i
    logical :: notlinked

    srcid = srcid_str%to_number(kind=1_i1p)
    notlinked = .true.  !assuming not linked
    do i=1, size(self%src)
        if (self%src(i)%par%id == srcid) then ! found the correct source to link to
            call self%src(i)%linkproperty(ptype,pname) ! calling Source method to link property
            temp = self%src(i)%par%id
            outext='      Source id = '// temp // ', '// self%src(i)%par%name //' is of type '// self%src(i)%par%property_type //', with property name ' // self%src(i)%par%property_name
            call log%put(outext,.false.)
            notlinked = .false. ! we linked it
            exit
        endif
    enddo
    if (notlinked) then ! property has no corresponding Source
        temp = srcid
        outext='      Source id = '// temp //' not listed, property '// pname //', of type ' // ptype // ' not linked, ignoring'
        call log%put(outext,.false.)
    endif
```

6.17.3 Variable Documentation

6.17.3.1 tempsources

type(source_group_class), public sources_mod::tempsources

Definition at line 82 of file sources.f90.

```
82      type(source_group_class) :: tempSources
```

6.18 tracer_array_mod Module Reference

Data Types

- type tracerarray

Functions/Subroutines

- subroutine print_tracerarray (this)
- subroutine print_tracerarray_element (this, index)

6.18.1 Function/Subroutine Documentation

6.18.1.1 print_tracerarray()

```
subroutine tracer_array_mod::print_tracerarray (
            class(tracerarray), intent(in) this )  [private]
```

Definition at line 36 of file tracer_array.f90.

```
36     class(TracerArray), intent(in) :: this
37     class(*), pointer :: curr
38     integer :: i
39     do i=1, this%usedLength
40         curr => this%get(i)
41         select type(curr)
42         type is (tracer_class)
43             !call curr%print()
44         class is (paper_class)
45             !call curr%print()
46         class is (plastic_class)
47             !call curr%print()
48             class default
49             stop ’[print_TracerArray]: unexepected type of content: not a shape or derived type’
50         end select
51     end do
```

6.18.1.2 print_tracerarray_element()

```
subroutine tracer_array_mod::print_tracerarray_element (
            class(tracerarray), intent(in) this,
            integer, intent(in) index )  [private]
```

Definition at line 55 of file tracer_array.f90.

```
55     class(TracerArray), intent(in) :: this
56     integer, intent(in) :: index
57     class(*), pointer :: curr
58     if (index .le. this%usedLength) then
59         curr => this%get(index)
60         select type(curr)
61         type is (tracer_class)
62             !call curr%print()
63         class is (paper_class)
64             !call curr%print()
65         class is (plastic_class)
66             !call curr%print()
67             class default
68             stop ’[print_TracerArray_Element]: unexepected type of content, not a shape or derived type’
69         end select
70     else
71         stop ’[print_TracerArray_Element]: index out of bounds’
72     endif
```

6.19 tracer_base_mod Module Reference

Module that defines a pure Lagrangian tracer class and related methods.

Data Types

- type tracer_class

  Type - The pure Lagrangian tracer class.
- type tracer_par_class
- type tracer_state_class

  Type - state variables of a pure Lagrangian tracer object.
- type tracer_stats_class

  Type - statistical variables of a pure Lagrangian tracer object.

Functions/Subroutines

- subroutine initialize (trc, id, id_source, time, pt)

  Tracer initialization method.

Variables

- type(tracer_class), dimension(:), allocatable, public tracer

6.19.1 Detailed Description

Module that defines a pure Lagrangian tracer class and related methods.

Author

Ricardo Birjukovs Canelas

6.19.2 Function/Subroutine Documentation

6.19.2.1 initialize()

```
subroutine tracer_base_mod::initialize (
             class(tracer_class) trc,
             integer, intent(in) id,
             integer, intent(in) id_source,
             real(prec_time), intent(in) time,
             type(vector), intent(in) pt )  [private]
```

Tracer initialization method.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in		

Definition at line 81 of file tracer_base.f90.

```
81     implicit none
82     class(tracer_class) :: trc
83     integer, intent(in) :: id
84     integer, intent(in) :: id_source
85     type(vector), intent(in) :: pt
86     real(prec_time), intent(in) :: time
87
88     ! initialize parameters
89     trc%par%id = id
90     trc%par%idsource = id_source
91     trc%par%velmax = 15.0 !(m/s, just a placeholder)
92     ! interp_method - TODO
93     ! initialize tracer state
94     trc%now%age=0.0
95     trc%now%active = .false.
96     trc%now%pos = pt
97     trc%now%vel = 0.0
98     trc%now%acc = 0.0
99     trc%now%depth = 0.0
100     ! Initialize statistical accumulator variables
101     trc%stats%acc_pos = 0.0
102     trc%stats%acc_vel = 0.0
103     trc%stats%acc_depth = 0.0
104     trc%stats%ns = 0
105
```

6.19.3 Variable Documentation

6.19.3.1 tracer

`type(tracer_class), dimension(:), allocatable, public tracer_base_mod::tracer`

Definition at line 64 of file tracer_base.f90.

```
64     type(tracer_class), allocatable, dimension(:) :: Tracer
```

6.20 tracer_interp_mod Module Reference

6.21 tracer_paper_mod Module Reference

Module that defines a Lagrangian tracer class for paper modelling and related methods. The type is defined as a derived type from the pule Lagrangian tracer, and hence inherits all of it's data and methods.

Data Types

- type paper_class

  Type - The plastic material Lagrangian tracer class.
- type paper_par_class
- type paper_state_class

  Type - State variables of a tracer object representing a paper material.

Functions/Subroutines

- subroutine paper_initialize (trc, id, id_source, time, pt)

  Tracer initialization method.

6.21.1 Detailed Description

Module that defines a Lagrangian tracer class for paper modelling and related methods. The type is defined as a derived type from the pule Lagrangian tracer, and hence inherits all of it's data and methods.

Author

Ricardo Birjukovs Canelas

6.21.2 Function/Subroutine Documentation

6.21.2.1 paper_initialize()

```
subroutine tracer_paper_mod::paper_initialize (
            class(paper_class) trc,
            integer, intent(in) id,
            integer, intent(in) id_source,
            real(prec_time), intent(in) time,
            type(vector), intent(in) pt )  [private]
```

Tracer initialization method.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in		

Definition at line 64 of file tracer_paper.f90.

```
64     implicit none
65     class(paper_class) :: trc
66     integer, intent(in) :: id
67     integer, intent(in) :: id_source
68     type(vector), intent(in) :: pt
69     real(prec_time), intent(in) :: time
70
71     ! initialize parameters
72     trc%par%id = id
73     trc%par%idsource = id_source
74     trc%par%velmax = 15.0 !(m/s, just a placeholder)
75     ! interp_method - TODO
76     ! initialize tracer state
77     trc%now%age=0.0
78     trc%now%active = .false.
```

```
    trc%now%pos = pt
    trc%now%vel = 0.0
    trc%now%acc = 0.0
    trc%now%depth = 0.0
    ! Initialize statistical accumulator variables
    trc%stats%acc_pos = 0.0
    trc%stats%acc_vel = 0.0
    trc%stats%acc_depth = 0.0
    trc%stats%ns = 0

```

6.22 tracer_plastic_mod Module Reference

Module that defines a Lagrangian tracer class for plastic modelling and related methods. The type is defined as a derived type from the pule Lagrangian tracer, and hence inherits all of it's data and methods.

Data Types

- type plastic_class

  Type - The plastic material Lagrangian tracer class.
- type plastic_par_class
- type plastic_state_class

  Type - State variables of a tracer object representing a plastic material.

Functions/Subroutines

- subroutine plastic_initialize (trc, id, id_source, time, pt)

  Tracer initialization method.

6.22.1 Detailed Description

Module that defines a Lagrangian tracer class for plastic modelling and related methods. The type is defined as a derived type from the pule Lagrangian tracer, and hence inherits all of it's data and methods.

Author

Ricardo Birjukovs Canelas

6.22.2 Function/Subroutine Documentation

6.22.2.1 plastic_initialize()

```
subroutine tracer_plastic_mod::plastic_initialize (
              class(plastic_class) trc,
              integer, intent(in) id,
              integer, intent(in) id_source,
              real(prec_time), intent(in) time,
              type(vector), intent(in) pt )  [private]
```

Tracer initialization method.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in		

Definition at line 64 of file tracer_plastic.f90.

```
    implicit none
    class(plastic_class) :: trc
    integer, intent(in) :: id
    integer, intent(in) :: id_source
    type(vector), intent(in) :: pt
    real(prec_time), intent(in) :: time

    ! initialize parameters
    trc%par%id = id
    trc%par%idsource = id_source
    trc%par%velmax = 15.0 !(m/s, just a placeholder)
    ! interp_method - TODO
    ! initialize tracer state
    trc%now%age=0.0
    trc%now%active = .false.
    trc%now%pos = pt
    trc%now%vel = 0.0
    trc%now%acc = 0.0
    trc%now%depth = 0.0
    ! Initialize statistical accumulator variables
    trc%stats%acc_pos = 0.0
    trc%stats%acc_vel = 0.0
    trc%stats%acc_depth = 0.0
    trc%stats%ns = 0

```

6.23 tracers_mod Module Reference

Module to hold and wrap all the tracer respective modules. Defines a pure Lagrangian tracer block. This is intended to serve as the base class for every type of tracer class needed, that should be built as derived of this class, with the necessary modifiers to model the desired behaviour. Basic tracer data (parameters, variables) are implemented. Tracer methods such as I/O, integration and interpolation routines are implemented.

6.23.1 Detailed Description

Module to hold and wrap all the tracer respective modules. Defines a pure Lagrangian tracer block. This is intended to serve as the base class for every type of tracer class needed, that should be built as derived of this class, with the necessary modifiers to model the desired behaviour. Basic tracer data (parameters, variables) are implemented. Tracer methods such as I/O, integration and interpolation routines are implemented.

Author

Ricardo Birjukovs Canelas

6.24 utilities_mod Module Reference

Module that provides useful back-end routines.

Functions/Subroutines

- real(prec) function, public get_closest_twopow (num)

  Public function that returns the closest power of 2 or a given real number.

6.24.1 Detailed Description

Module that provides useful back-end routines.

Author

Ricardo Birjukovs Canelas

6.24.2 Function/Subroutine Documentation

6.24.2.1 get_closest_twopow()

```
real(prec) function, public utilities_mod::get_closest_twopow (
            real(prec), intent(in) num )
```

Public function that returns the closest power of 2 or a given real number.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*num*	

Definition at line 45 of file utilities.f90.

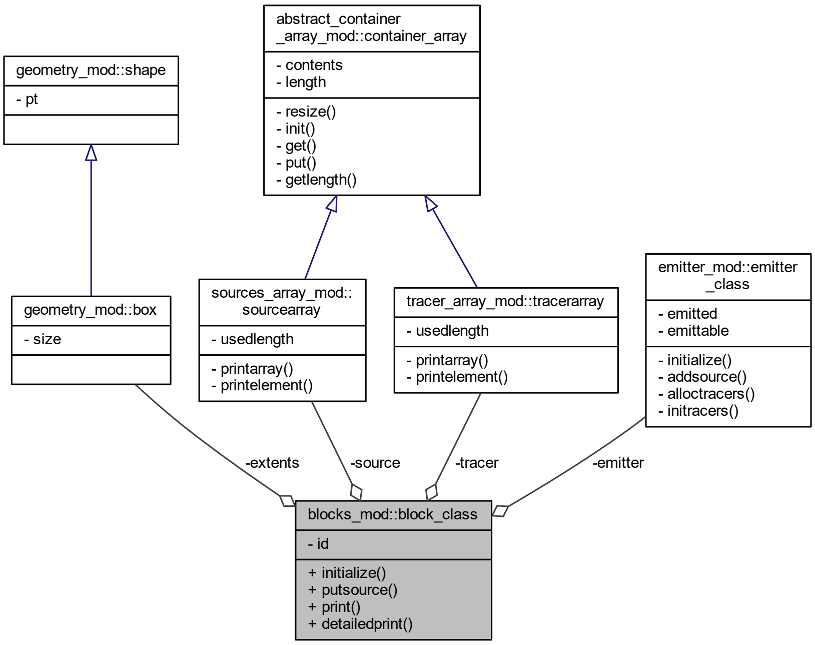
```
45     implicit none
46     real(prec), intent(in) :: num
47     real(prec) :: twopow
48     integer :: i
49     real(prec) :: dist1, dist2
50
51     do i=-4, 10
52         twopow = 2.0**i
53         if (num < twopow) then
54             dist1 = sqrt(twopow-num)
55             dist2 = sqrt(num-2.0**(i-1))
56             if (dist2 < dist1) then
57                 twopow = 2.0**(i-1)
58                 exit
59             endif
60             exit
61         endif
62     enddo
63
```

Chapter 7

Data Type Documentation

7.1 blocks_mod::block_class Type Reference

Collaboration diagram for blocks_mod::block_class:



Public Member Functions

- procedure, public initialize => initBlock
- procedure, public putsource
- procedure, public print => printBlock
- procedure, public detailedprint => printdetailBlock

Private Attributes

- integer id
- type(box) extents

    shape::box that defines the extents of this block
- type(sourcearray) source

    List of Sources currently on this block.
- type(tracerarray) tracer

    List of Tracers currently on this block.
- type(emitter_class) emitter

    Block Emitter.

7.1.1 Detailed Description

Definition at line 35 of file blocks.f90.

7.1.2 Member Function/Subroutine Documentation

7.1.2.1 detailedprint()

```
procedure, public blocks_mod::block_class::detailedprint ( )
```

Definition at line 46 of file blocks.f90.

7.1.2.2 initialize()

```
procedure, public blocks_mod::block_class::initialize ( )
```

Definition at line 43 of file blocks.f90.

7.1.2.3 print()

```
procedure, public blocks_mod::block_class::print ( )
```

Definition at line 45 of file blocks.f90.

7.1.2.4 putsource()

```
procedure, public blocks_mod::block_class::putsource ( )
```

Definition at line 44 of file blocks.f90.

7.1.3 Member Data Documentation

7.1.3.1 emitter

```
type(emitter_class) blocks_mod::block_class::emitter  [private]
```

Block Emitter.

Definition at line 40 of file blocks.f90.

```
40         type(emitter_class) :: Emitter
```

7.1.3.2 extents

```
type(box) blocks_mod::block_class::extents  [private]
```

shape::box that defines the extents of this block

Definition at line 37 of file blocks.f90.

```
37         type(box) :: extents
```

7.1.3.3 id

```
integer blocks_mod::block_class::id  [private]
```

Definition at line 36 of file blocks.f90.

```
36         integer :: id
```

7.1.3.4 source

```
type(sourcearray) blocks_mod::block_class::source  [private]
```

List of Sources currently on this block.

Definition at line 38 of file blocks.f90.

```
        type(SourceArray) :: Source
```

7.1.3.5 tracer

```
type(tracerarray) blocks_mod::block_class::tracer  [private]
```

List of Tracers currently on this block.

Definition at line 39 of file blocks.f90.

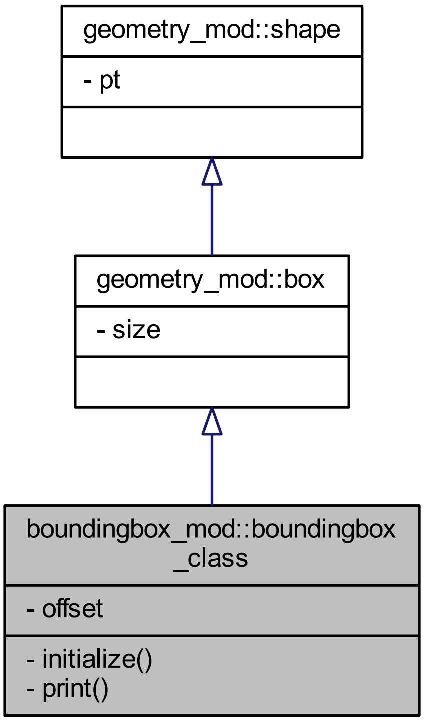
```
        type(TracerArray) :: Tracer
```

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

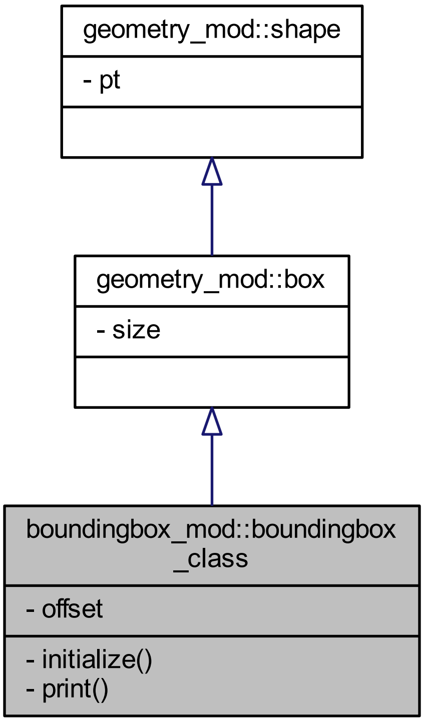
- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/blocks.f90

7.2 boundingbox_mod::boundingbox_class Type Reference

Inheritance diagram for boundingbox_mod::boundingbox_class:



Collaboration diagram for boundingbox_mod::boundingbox_class:



Private Member Functions

- procedure initialize => initboundingbox
- procedure print => printboundingbox

Private Attributes

- type(vector) offset

7.2.1 Detailed Description

Definition at line 26 of file boundingbox.f90.

7.2.2 Member Function/Subroutine Documentation

7.2.2.1 initialize()

```
procedure boundingbox_mod::boundingbox_class::initialize ( )  [private]
```

Definition at line 29 of file boundingbox.f90.

7.2.2.2 print()

```
procedure boundingbox_mod::boundingbox_class::print ( )  [private]
```

Definition at line 30 of file boundingbox.f90.

7.2.3 Member Data Documentation

7.2.3.1 offset

```
type(vector) boundingbox_mod::boundingbox_class::offset  [private]
```

Definition at line 27 of file boundingbox.f90.

```
27   type(vector) :: offset
```

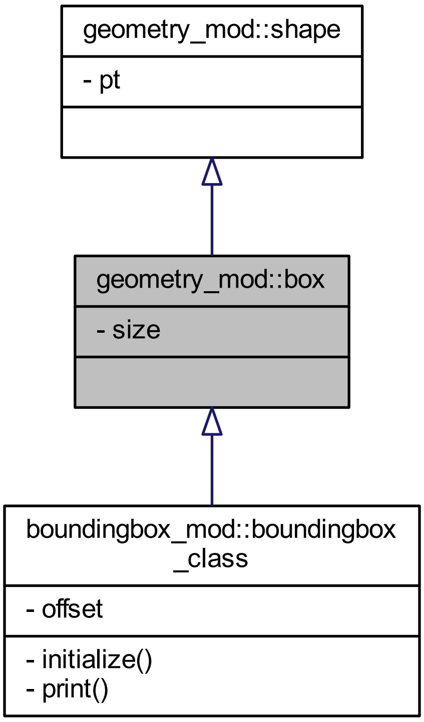
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/boundingbox.f90

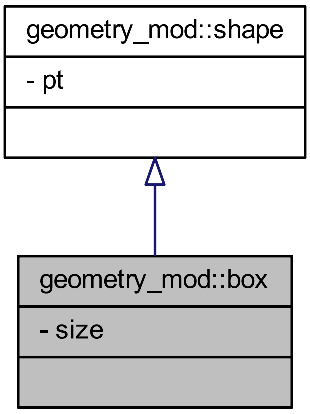
7.3 geometry_mod::box Type Reference

Type - point class.

Inheritance diagram for geometry_mod::box:



Collaboration diagram for geometry_mod::box:



Private Attributes

- type(vector) size

    Box size.

7.3.1 Detailed Description

Type - point class.

Definition at line 57 of file geometry.f90.

7.3.2 Member Data Documentation

7.3.2.1 size

```
type(vector) geometry_mod::box::size  [private]
```

Box size.

Definition at line 58 of file geometry.f90.

```
58          type(vector) :: size
```

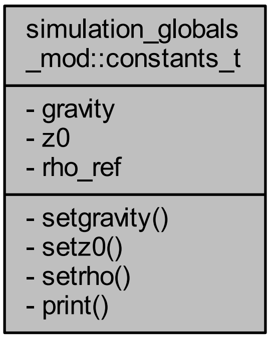
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/geometry.f90

7.4 simulation_globals_mod::constants_t Type Reference

Case Constants class.

Collaboration diagram for simulation_globals_mod::constants_t:



Private Member Functions

- procedure setgravity
- procedure setz0
- procedure setrho
- procedure print => printconstants

Private Attributes

- type(vector) gravity

  Gravitational acceleration vector (default=(0 0 -9.81)) (m s-2)
- real(prec) z0 = 0.0

  Reference local sea level.
- real(prec) rho_ref = 1000.0

  Reference density of the medium (default=1000.0) (kg m-3)

7.4.1 Detailed Description

Case Constants class.

Definition at line 61 of file simulation_globals.f90.

7.4.2 Member Function/Subroutine Documentation

7.4.2.1 print()

```
procedure simulation_globals_mod::constants_t::print ( )  [private]
```

Definition at line 69 of file simulation_globals.f90.

7.4.2.2 setgravity()

```
procedure simulation_globals_mod::constants_t::setgravity ( )  [private]
```

Definition at line 66 of file simulation_globals.f90.

7.4.2.3 setrho()

```
procedure simulation_globals_mod::constants_t::setrho ( )  [private]
```

Definition at line 68 of file simulation_globals.f90.

7.4.2.4 setz0()

```
procedure simulation_globals_mod::constants_t::setz0 ( )  [private]
```

Definition at line 67 of file simulation_globals.f90.

7.4.3 Member Data Documentation

7.4.3.1 gravity

```
type(vector) simulation_globals_mod::constants_t::gravity  [private]
```

Gravitational acceleration vector (default=(0 0 -9.81)) (m s-2)

Definition at line 62 of file simulation_globals.f90.

```
62         type(vector) :: Gravity
```

7.4.3.2 rho_ref

```
real(prec) simulation_globals_mod::constants_t::rho_ref = 1000.0  [private]
```

Reference density of the medium (default=1000.0) (kg m-3)

Definition at line 64 of file simulation_globals.f90.

```
64         real(prec)   :: Rho_ref = 1000.0
```

7.4.3.3 z0

```
real(prec) simulation_globals_mod::constants_t::z0 = 0.0  [private]
```

Reference local sea level.

Definition at line 63 of file simulation_globals.f90.

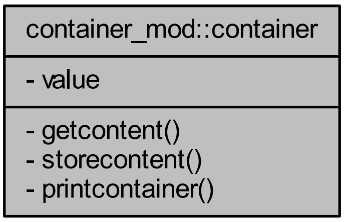
```
63         real(prec)   :: Z0 = 0.0
```

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

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/simulation_globals.f90

7.5 container_mod::container Interface Reference

Collaboration diagram for container_mod::container:



Private Member Functions

- procedure getcontent

  returns stored content (pointer)
- procedure storecontent

  stores the provided values (sourced allocation)
- procedure printcontainer

  prints container contents (only primitive types implemented)

Private Attributes

- class(∗), pointer value => null()

  value stored in container

7.5.1 Detailed Description

Definition at line 40 of file container.f90.

7.5.2 Member Function/Subroutine Documentation

7.5.2.1 getcontent()

```
procedure container_mod::container::getcontent ( )  [private]
```

returns stored content (pointer)

Definition at line 44 of file container.f90.

7.5.2.2 printcontainer()

```
procedure container_mod::container::printcontainer ( )  [private]
```

prints container contents (only primitive types implemented)

Definition at line 46 of file container.f90.

7.5.2.3 storecontent()

```
procedure container_mod::container::storecontent ( )  [private]
```

stores the provided values (sourced allocation)

Definition at line 45 of file container.f90.

7.5.3 Member Data Documentation

7.5.3.1 value

```
class(*), pointer container_mod::container::value => null()  [private]
```

value stored in container

Definition at line 42 of file container.f90.

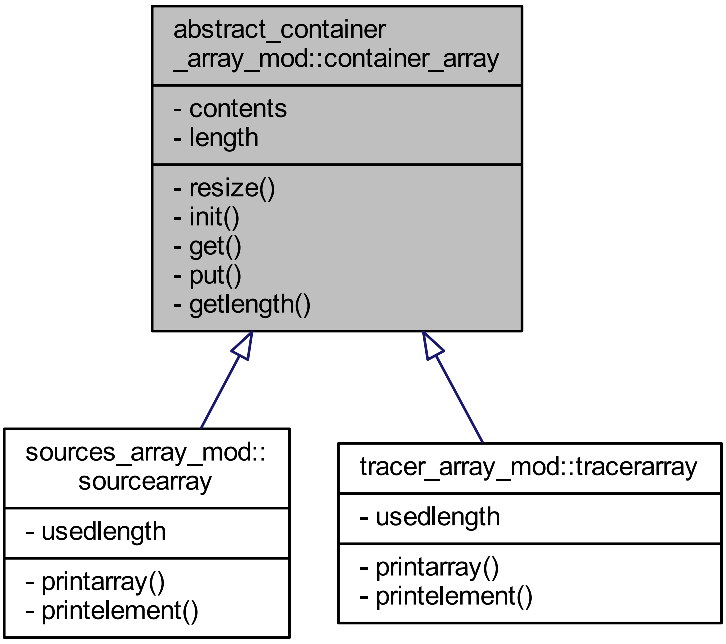
```
42         class(*), pointer :: value => null()
```

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

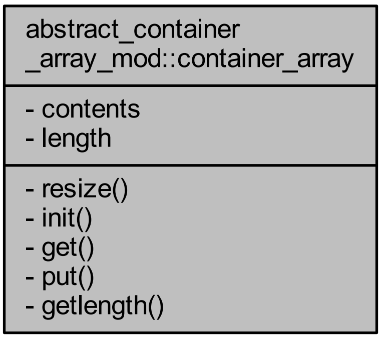
- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/container.f90

7.6 abstract_container_array_mod::container_array Type Reference

Inheritance diagram for abstract_container_array_mod::container_array:



Collaboration diagram for abstract_container_array_mod::container_array:



Private Member Functions

- procedure resize => resizeArray

Grows (adds empty space) or shrinks (discards the last entries) of the array.

- procedure init => initArray

  Allocates the container array. Deallocates if already allocated.

- procedure, non_overridable get => getValue

  returns the requested entry (pointer)

- procedure, non_overridable put => putValue

  stores a value on the requested index

- procedure, non_overridable getlength

  returns the length of the array

Private Attributes

- class(container), dimension(:), allocatable contents

  Allocatable unlimited polymorphic container array.

- integer length

  Lenght of the array, for easy access.

7.6.1 Detailed Description

Definition at line 44 of file abstract_container_array.f90.

7.6.2 Member Function/Subroutine Documentation

7.6.2.1 get()

```
procedure, non_overridable abstract_container_array_mod::container_array::get ( ) [private]
```

returns the requested entry (pointer)

Definition at line 51 of file abstract_container_array.f90.

7.6.2.2 getlength()

```
procedure, non_overridable abstract_container_array_mod::container_array::getlength ( ) [private]
```

returns the length of the array

Definition at line 53 of file abstract_container_array.f90.

7.6.2.3 init()

```
procedure abstract_container_array_mod::container_array::init ( )  [private]
```

Allocates the container array. Deallocates if already allocated.

Definition at line 50 of file abstract_container_array.f90.

7.6.2.4 put()

```
procedure, non_overridable abstract_container_array_mod::container_array::put ( )  [private]
```

stores a value on the requested index

Definition at line 52 of file abstract_container_array.f90.

7.6.2.5 resize()

```
procedure abstract_container_array_mod::container_array::resize ( )  [private]
```

Grows (adds empty space) or shrinks (discards the last entries) of the array.

Definition at line 49 of file abstract_container_array.f90.

7.6.3 Member Data Documentation

7.6.3.1 contents

```
class(container), dimension(:), allocatable abstract_container_array_mod::container_array↩
::contents  [private]
```

Allocatable unlimited polymorphic container array.

Definition at line 46 of file abstract_container_array.f90.

```
        class(container), allocatable, dimension(:) :: contents
```

7.6.3.2 length

```
integer abstract_container_array_mod::container_array::length  [private]
```

Lenght of the array, for easy access.

Definition at line 47 of file abstract_container_array.f90.

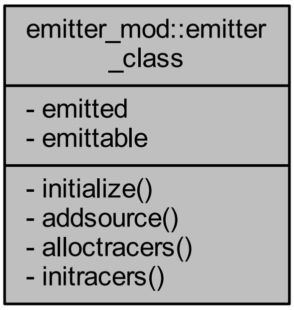
```
47         integer :: length
```

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

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/abstract_container_array.f90

7.7 emitter_mod::emitter_class Type Reference

Collaboration diagram for emitter_mod::emitter_class:



Private Member Functions

- procedure initialize => initializeEmitter
- procedure addsource
- procedure alloctracers
- procedure initracers

Private Attributes

- integer emitted
- integer emittable

7.7.1 Detailed Description

Definition at line 30 of file emitter.f90.

7.7.2 Member Function/Subroutine Documentation

7.7.2.1 addsource()

```
procedure emitter_mod::emitter_class::addsource ( )  [private]
```

Definition at line 35 of file emitter.f90.

7.7.2.2 alloctracers()

```
procedure emitter_mod::emitter_class::alloctracers ( )  [private]
```

Definition at line 36 of file emitter.f90.

7.7.2.3 initialize()

```
procedure emitter_mod::emitter_class::initialize ( )  [private]
```

Definition at line 34 of file emitter.f90.

7.7.2.4 initracers()

```
procedure emitter_mod::emitter_class::initracers ( )  [private]
```

Definition at line 37 of file emitter.f90.

7.7.3 Member Data Documentation

7.7.3.1 emittable

```
integer emitter_mod::emitter_class::emittable  [private]
```

Definition at line 32 of file emitter.f90.

```
32        integer :: emittable
```

7.7.3.2 emitted

```
integer emitter_mod::emitter_class::emitted  [private]
```

Definition at line 31 of file emitter.f90.

```
31        integer :: emitted
```

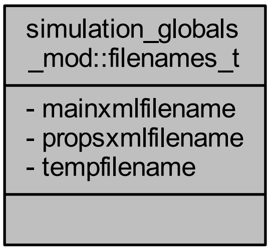
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/emitter.f90

7.8 simulation_globals_mod::filenames_t Type Reference

File names class.

Collaboration diagram for simulation_globals_mod::filenames_t:



Private Attributes

- type(string) mainxmlfilename

  Input .xml file name.
- type(string) propsxmlfilename

  Properties .xml file name.
- type(string) tempfilename

  Generic temporary file name.

7.8.1 Detailed Description

File names class.

Definition at line 72 of file simulation_globals.f90.

7.8.2 Member Data Documentation

7.8.2.1 mainxmlfilename

```
type(string) simulation_globals_mod::filenames_t::mainxmlfilename [private]
```

Input .xml file name.

Definition at line 73 of file simulation_globals.f90.

```
73         type(string) :: mainxmlfilename
```

7.8.2.2 propsxmlfilename

```
type(string) simulation_globals_mod::filenames_t::propsxmlfilename [private]
```

Properties .xml file name.

Definition at line 74 of file simulation_globals.f90.

```
74         type(string) :: propsxmlfilename
```

7.8.2.3 tempfilename

```
type(string) simulation_globals_mod::filenames_t::tempfilename [private]
```

Generic temporary file name.

Definition at line 75 of file simulation_globals.f90.

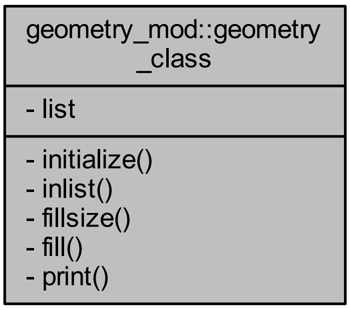
```
75         type(string) :: tempfilename
```

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

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/simulation_globals.f90

7.9 geometry_mod::geometry_class Type Reference

Collaboration diagram for geometry_mod::geometry_class:



Private Member Functions

- procedure initialize => allocatelist

  Builds the geometry list, possible geometry types (new types must be manually added)
- procedure inlist

  checks if a given geometry is defined as a derived type (new types must be manually added)
- procedure fillsize

  Gets the number of points that fill a geometry (based on GLOBALS::dp)
- procedure fill

  Gets the list of points that fill a geometry (based on GLOBALS::dp)
- procedure print => printGeometry

  prints the geometry type and contents

Private Attributes

- type(string), dimension(:), allocatable list

  String list (array) with the name of possible geometry types.

7.9.1 Detailed Description

Definition at line 32 of file geometry.f90.

7.9.2 Member Function/Subroutine Documentation

7.9.2.1 fill()

```
procedure geometry_mod::geometry_class::fill ( )  [private]
```

Gets the list of points that fill a geometry (based on GLOBALS::dp)

Definition at line 38 of file geometry.f90.

7.9.2.2 fillsize()

```
procedure geometry_mod::geometry_class::fillsize ( )  [private]
```

Gets the number of points that fill a geometry (based on GLOBALS::dp)

Definition at line 37 of file geometry.f90.

7.9.2.3 initialize()

```
procedure geometry_mod::geometry_class::initialize ( )  [private]
```

Builds the geometry list, possible geometry types (new types must be manually added)

Definition at line 35 of file geometry.f90.

7.9.2.4 inlist()

```
procedure geometry_mod::geometry_class::inlist ( )  [private]
```

checks if a given geometry is defined as a derived type (new types must be manually added)

Definition at line 36 of file geometry.f90.

7.9.2.5 print()

```
procedure geometry_mod::geometry_class::print ( )  [private]
```

prints the geometry type and contents

Definition at line 39 of file geometry.f90.

7.9.3 Member Data Documentation

7.9.3.1 list

```
type(string), dimension(:), allocatable geometry_mod::geometry_class::list  [private]
```

String list (array) with the name of possible geometry types.

Definition at line 33 of file geometry.f90.

```
33         type(string), allocatable, dimension(:) :: list
```

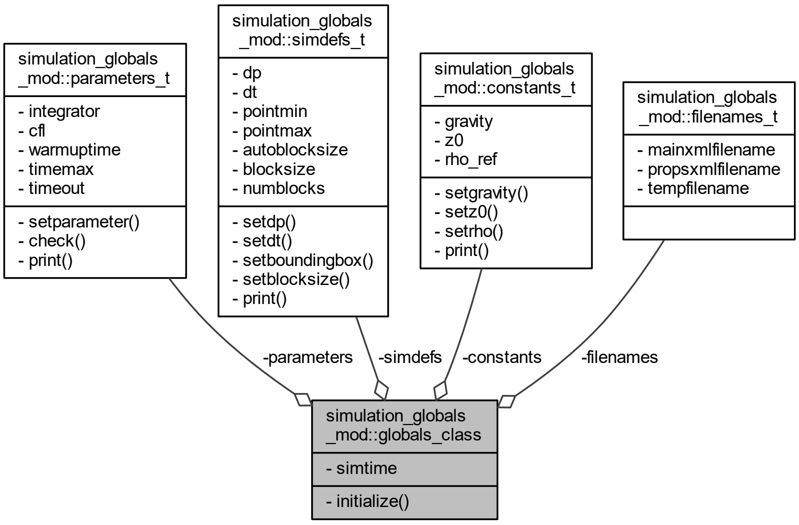
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/geometry.f90

7.10 simulation_globals_mod::globals_class Type Reference

Globals class - This is a container for every global variable on the simulation.

Collaboration diagram for simulation_globals_mod::globals_class:



Private Member Functions

- procedure initialize => setdefaults

Private Attributes

- type(parameters_t) parameters
- type(simdefs_t) simdefs
- type(constants_t) constants
- type(filenames_t) filenames
- real(prec_time) simtime

7.10.1 Detailed Description

Globals class - This is a container for every global variable on the simulation.

Definition at line 78 of file simulation_globals.f90.

7.10.2 Member Function/Subroutine Documentation

7.10.2.1 initialize()

```
procedure simulation_globals_mod::globals_class::initialize ( )  [private]
```

Definition at line 85 of file simulation_globals.f90.

7.10.3 Member Data Documentation

7.10.3.1 constants

```
type(constants_t) simulation_globals_mod::globals_class::constants  [private]
```

Definition at line 81 of file simulation_globals.f90.

```
81         type(constants_t)   :: Constants
```

7.10.3.2 filenames

```
type(filenames_t) simulation_globals_mod::globals_class::filenames  [private]
```

Definition at line 82 of file simulation_globals.f90.

```
82         type(filenames_t)   :: FileNames
```

7.10.3.3 parameters

```
type(parameters_t) simulation_globals_mod::globals_class::parameters  [private]
```

Definition at line 79 of file simulation_globals.f90.

```
79        type(parameters_t)  :: Parameters
```

7.10.3.4 simdefs

```
type(simdefs_t) simulation_globals_mod::globals_class::simdefs  [private]
```

Definition at line 80 of file simulation_globals.f90.

```
80        type(simdefs_t)     :: SimDefs
```

7.10.3.5 simtime

```
real(prec_time) simulation_globals_mod::globals_class::simtime  [private]
```

Definition at line 83 of file simulation_globals.f90.

```
83        real(prec_time)     :: SimTime
```

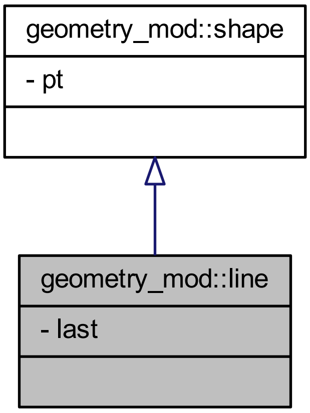
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/simulation_globals.f90

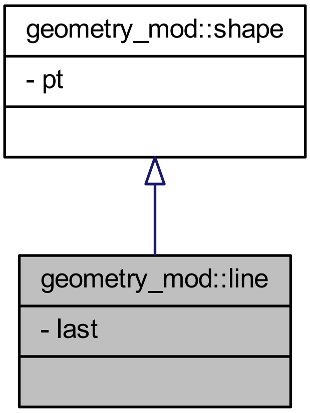
7.11 geometry_mod::line Type Reference

Type - line class.

Inheritance diagram for geometry_mod::line:



Collaboration diagram for geometry_mod::line:



Private Attributes

- type(vector) last

  Coordinates of the end point.

7.11.1 Detailed Description

Type - line class.

Definition at line 49 of file geometry.f90.

7.11.2 Member Data Documentation

7.11.2.1 last

```
type(vector) geometry_mod::line::last  [private]
```

Coordinates of the end point.

Definition at line 50 of file geometry.f90.

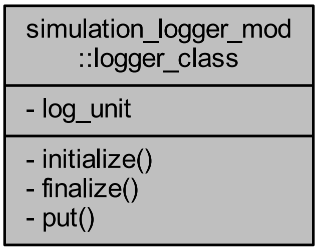
```
50        type(vector) :: last
```

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

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/geometry.f90

7.12 simulation_logger_mod::logger_class Type Reference

Collaboration diagram for simulation_logger_mod::logger_class:



Private Member Functions

- procedure initialize => initLog
- procedure finalize => closeLog
- procedure put => put_inLog

Private Attributes

- integer log_unit = -1

7.12.1 Detailed Description

Definition at line 29 of file simulation_logger.f90.

7.12.2 Member Function/Subroutine Documentation

7.12.2.1 finalize()

```
procedure simulation_logger_mod::logger_class::finalize ( )  [private]
```

Definition at line 34 of file simulation_logger.f90.

7.12.2.2 initialize()

```
procedure simulation_logger_mod::logger_class::initialize ( )  [private]
```

Definition at line 33 of file simulation_logger.f90.

7.12.2.3 put()

```
procedure simulation_logger_mod::logger_class::put ( )  [private]
```

Definition at line 35 of file simulation_logger.f90.

7.12.3 Member Data Documentation

7.12.3.1 log_unit

```
integer simulation_logger_mod::logger_class::log_unit = -1  [private]
```

Definition at line 31 of file simulation_logger.f90.

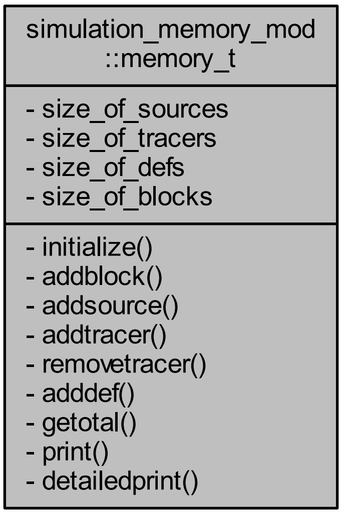
```
31         integer :: log_unit = -1
```

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

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/simulation_logger.f90

7.13 simulation_memory_mod::memory_t Type Reference

Collaboration diagram for simulation_memory_mod::memory_t:



Private Member Functions

- procedure initialize => initializeMemory
- procedure addblock
- procedure addsource
- procedure addtracer
- procedure removetracer
- procedure adddef
- procedure getotal
- procedure print => printmemory
- procedure detailedprint => printmemorydetailed

Private Attributes

- integer size_of_sources

  Size of the sources in memory (bytes)
- integer size_of_tracers

  Size of the tracers in memory (bytes)
- integer size_of_defs

  Size of the parameters and definitions in memory (bytes)
- integer size_of_blocks

  Size of the Blocks in memory (bytes)

7.13.1 Detailed Description

Definition at line 28 of file simulation_memory.f90.

7.13.2 Member Function/Subroutine Documentation

7.13.2.1 addblock()

```
procedure simulation_memory_mod::memory_t::addblock ( )  [private]
```

Definition at line 35 of file simulation_memory.f90.

7.13.2.2 adddef()

```
procedure simulation_memory_mod::memory_t::adddef ( )  [private]
```

Definition at line 39 of file simulation_memory.f90.

7.13.2.3 addsource()

```
procedure simulation_memory_mod::memory_t::addsource ( )  [private]
```

Definition at line 36 of file simulation_memory.f90.

7.13.2.4 addtracer()

```
procedure simulation_memory_mod::memory_t::addtracer ( )  [private]
```

Definition at line 37 of file simulation_memory.f90.

7.13.2.5 detailedprint()

```
procedure simulation_memory_mod::memory_t::detailedprint ( )  [private]
```

Definition at line 42 of file simulation_memory.f90.

7.13.2.6 getotal()

```
procedure simulation_memory_mod::memory_t::getotal ( )  [private]
```

Definition at line 40 of file simulation_memory.f90.

7.13.2.7 initialize()

```
procedure simulation_memory_mod::memory_t::initialize ( )  [private]
```

Definition at line 34 of file simulation_memory.f90.

7.13.2.8 print()

```
procedure simulation_memory_mod::memory_t::print ( )  [private]
```

Definition at line 41 of file simulation_memory.f90.

7.13.2.9 removetracer()

```
procedure simulation_memory_mod::memory_t::removetracer ( )  [private]
```

Definition at line 38 of file simulation_memory.f90.

7.13.3 Member Data Documentation

7.13.3.1 size_of_blocks

```
integer simulation_memory_mod::memory_t::size_of_blocks  [private]
```

Size of the Blocks in memory (bytes)

Definition at line 32 of file simulation_memory.f90.

```
32         integer :: size_of_blocks
```

7.13.3.2 size_of_defs

```
integer simulation_memory_mod::memory_t::size_of_defs  [private]
```

Size of the parameters and definitions in memory (bytes)

Definition at line 31 of file simulation_memory.f90.

```
31         integer :: size_of_defs
```

7.13.3.3 size_of_sources

```
integer simulation_memory_mod::memory_t::size_of_sources  [private]
```

Size of the sources in memory (bytes)

Definition at line 29 of file simulation_memory.f90.

```
29         integer :: size_of_sources
```

7.13.3.4 size_of_tracers

```
integer simulation_memory_mod::memory_t::size_of_tracers  [private]
```

Size of the tracers in memory (bytes)

Definition at line 30 of file simulation_memory.f90.

```
30         integer :: size_of_tracers
```

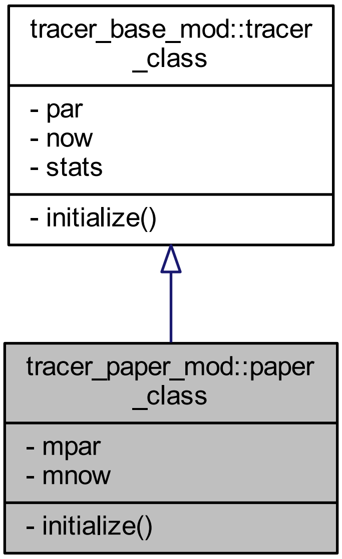
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/simulation_memory.f90

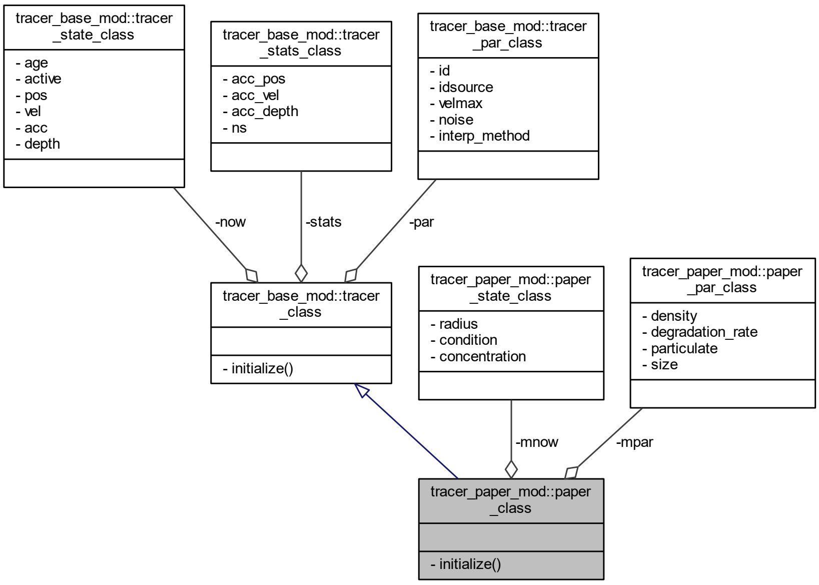
7.14 tracer_paper_mod::paper_class Type Reference

Type - The plastic material Lagrangian tracer class.

Inheritance diagram for tracer_paper_mod::paper_class:



Collaboration diagram for tracer_paper_mod::paper_class:



Private Member Functions

- procedure initialize => paper_initialize

Private Attributes

- type(paper_par_class) mpar

  To access material parameters.
- type(paper_state_class) mnow

  To access material state variables.

7.14.1 Detailed Description

Type - The plastic material Lagrangian tracer class.

Definition at line 42 of file tracer_paper.f90.

7.14.2 Member Function/Subroutine Documentation

7.14.2.1 initialize()

```
procedure tracer_paper_mod::paper_class::initialize ( )  [private]
```

Definition at line 46 of file tracer_paper.f90.

7.14.3 Member Data Documentation

7.14.3.1 mnow

```
type(paper_state_class) tracer_paper_mod::paper_class::mnow  [private]
```

To access material state variables.

Definition at line 44 of file tracer_paper.f90.

```
44         type(paper_state_class) :: mnow
```

7.14.3.2 mpar

```
type(paper_par_class) tracer_paper_mod::paper_class::mpar  [private]
```

To access material parameters.

Definition at line 43 of file tracer_paper.f90.

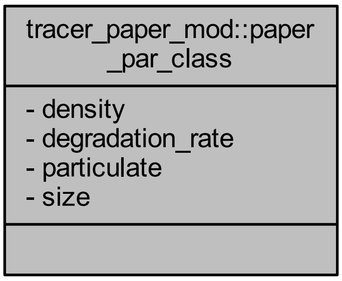
```
43        type(paper_par_class)   :: mpar
```

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

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracer_paper.f90

7.15 tracer_paper_mod::paper_par_class Type Reference

Collaboration diagram for tracer_paper_mod::paper_par_class:



Private Attributes

- real(prec) density

    density of the material
- real(prec) degradation_rate

    degradation rate of the material
- logical particulate

    flag to indicate if the material is a particle (false) or a collection of particles (true)
- real(prec) size

    Size (radius) of the particles (equals to the tracer radius if particulate==false)

7.15.1 Detailed Description

Definition at line 29 of file tracer_paper.f90.

7.15.2 Member Data Documentation

7.15.2.1 degradation_rate

```
real(prec) tracer_paper_mod::paper_par_class::degradation_rate  [private]
```

degradation rate of the material

Definition at line 31 of file tracer_paper.f90.

```
31         real(prec) :: degradation_rate
```

7.15.2.2 density

```
real(prec) tracer_paper_mod::paper_par_class::density  [private]
```

density of the material

Definition at line 30 of file tracer_paper.f90.

```
30         real(prec) :: density
```

7.15.2.3 particulate

```
logical tracer_paper_mod::paper_par_class::particulate  [private]
```

flag to indicate if the material is a particle (false) or a collection of particles (true)

Definition at line 32 of file tracer_paper.f90.

```
32         logical    :: particulate
```

7.15.2.4 size

```
real(prec) tracer_paper_mod::paper_par_class::size  [private]
```

Size (radius) of the particles (equals to the tracer radius if particulate==false)

Definition at line 33 of file tracer_paper.f90.

```
33         real(prec) :: size
```

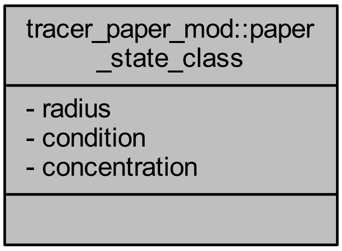
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracer_paper.f90

7.16 tracer_paper_mod::paper_state_class Type Reference

Type - State variables of a tracer object representing a paper material.

Collaboration diagram for tracer_paper_mod::paper_state_class:



Private Attributes

- real(prec) radius

  Tracer radius (m)
- real(prec) condition

  Material condition (1-0)
- real(prec) concentration

  Particle concentration.

7.16.1 Detailed Description

Type - State variables of a tracer object representing a paper material.

Definition at line 36 of file tracer_paper.f90.

7.16.2 Member Data Documentation

7.16.2.1 concentration

```
real(prec) tracer_paper_mod::paper_state_class::concentration  [private]
```

Particle concentration.

Definition at line 39 of file tracer_paper.f90.

```
39         real(prec) :: concentration
```

7.16.2.2 condition

```
real(prec) tracer_paper_mod::paper_state_class::condition  [private]
```

Material condition (1-0)

Definition at line 38 of file tracer_paper.f90.

```
38         real(prec) :: condition
```

7.16.2.3 radius

```
real(prec) tracer_paper_mod::paper_state_class::radius  [private]
```

Tracer radius (m)

Definition at line 37 of file tracer_paper.f90.

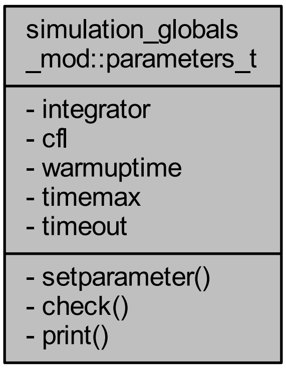
```
37         real(prec) :: radius
```

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

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracer_paper.f90

7.17 simulation_globals_mod::parameters_t Type Reference

Collaboration diagram for simulation_globals_mod::parameters_t:



Private Member Functions

- procedure setparameter
- procedure check
- procedure print => printsimparameters

Private Attributes

- integer integrator = 1

  Integration Algorithm 1:Verlet, 2:Symplectic, 3:RK4 (default=1)
- real(prec) cfl = 0.5

  Courant Friedrichs Lewy condition number.
- real(prec) warmuptime = 0.0

  Time to freeze the tracers at simulation start (warmup) (s) (default=0.0)
- real(prec) timemax = MV

  Simulation duration (s)
- real(prec) timeout = MV

  Time out data (1/Hz)

7.17.1 Detailed Description

Definition at line 33 of file simulation_globals.f90.

7.17.2 Member Function/Subroutine Documentation

7.17.2.1 check()

```
procedure simulation_globals_mod::parameters_t::check ( )  [private]
```

Definition at line 41 of file simulation_globals.f90.

7.17.2.2 print()

```
procedure simulation_globals_mod::parameters_t::print ( )  [private]
```

Definition at line 42 of file simulation_globals.f90.

7.17.2.3 setparameter()

```
procedure simulation_globals_mod::parameters_t::setparameter ( )  [private]
```

Definition at line 40 of file simulation_globals.f90.

7.17.3 Member Data Documentation

7.17.3.1 cfl

```
real(prec) simulation_globals_mod::parameters_t::cfl = 0.5  [private]
```

Courant Friedrichs Lewy condition number.

Definition at line 35 of file simulation_globals.f90.

```
35         real(prec) :: CFL = 0.5
```

7.17.3.2 integrator

```
integer simulation_globals_mod::parameters_t::integrator = 1  [private]
```

Integration Algorithm 1:Verlet, 2:Symplectic, 3:RK4 (default=1)

Definition at line 34 of file simulation_globals.f90.

```
34         integer    :: Integrator = 1
```

7.17.3.3 timemax

```
real(prec) simulation_globals_mod::parameters_t::timemax = MV  [private]
```

Simulation duration (s)

Definition at line 37 of file simulation_globals.f90.

```
37         real(prec) :: TimeMax = mv
```

7.17.3.4 timeout

```
real(prec) simulation_globals_mod::parameters_t::timeout = MV  [private]
```

Time out data (1/Hz)

Definition at line 38 of file simulation_globals.f90.

```
38         real(prec) :: TimeOut = mv
```

7.17.3.5 warmuptime

```
real(prec) simulation_globals_mod::parameters_t::warmuptime = 0.0  [private]
```

Time to freeze the tracers at simulation start (warmup) (s) (default=0.0)

Definition at line 36 of file simulation_globals.f90.

```
36         real(prec) :: WarmUpTime = 0.0
```

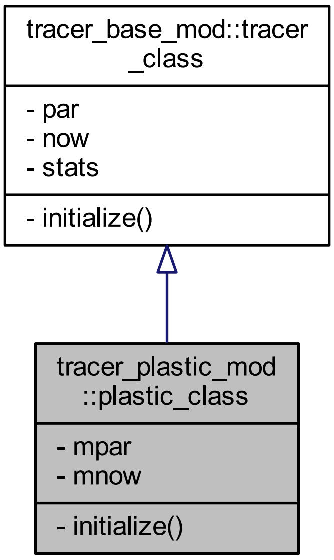
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/simulation_globals.f90

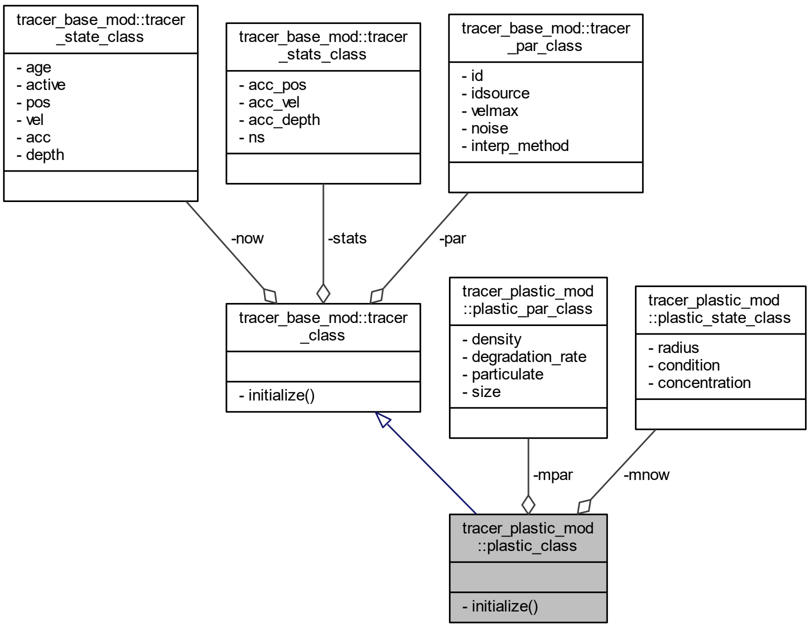
7.18 tracer_plastic_mod::plastic_class Type Reference

Type - The plastic material Lagrangian tracer class.

Inheritance diagram for tracer_plastic_mod::plastic_class:


tracer_base_mod::tracer_class
- par
- now
- stats
- initialize()
tracer_plastic_mod::plastic_class
- mpar
- mnow
- initialize()

Collaboration diagram for tracer_plastic_mod::plastic_class:



Private Member Functions

- procedure initialize => plastic_initialize

Private Attributes

- type(plastic_par_class) mpar

  To access material parameters.
- type(plastic_state_class) mnow

  To access material state variables.

7.18.1 Detailed Description

Type - The plastic material Lagrangian tracer class.

Definition at line 42 of file tracer_plastic.f90.

7.18.2 Member Function/Subroutine Documentation

7.18.2.1 initialize()

```
procedure tracer_plastic_mod::plastic_class::initialize ( )  [private]
```

Definition at line 46 of file tracer_plastic.f90.

7.18.3 Member Data Documentation

7.18.3.1 mnow

```
type(plastic_state_class) tracer_plastic_mod::plastic_class::mnow  [private]
```

To access material state variables.

Definition at line 44 of file tracer_plastic.f90.

```
44         type(plastic_state_class) :: mnow
```

7.18.3.2 mpar

```
type(plastic_par_class) tracer_plastic_mod::plastic_class::mpar  [private]
```

To access material parameters.

Definition at line 43 of file tracer_plastic.f90.

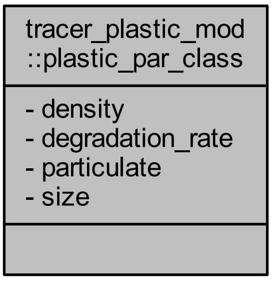
```
43         type(plastic_par_class)   :: mpar
```

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

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracer_plastic.f90

7.19 tracer_plastic_mod::plastic_par_class Type Reference

Collaboration diagram for tracer_plastic_mod::plastic_par_class:



Private Attributes

- real(prec) density

  density of the material
- real(prec) degradation_rate

  degradation rate of the material
- logical particulate

  flag to indicate if the material is a particle (false) or a collection of particles (true)
- real(prec) size

  Size (radius) of the particles (equals to the tracer radius if particulate==false)

7.19.1 Detailed Description

Definition at line 29 of file tracer_plastic.f90.

7.19.2 Member Data Documentation

7.19.2.1 degradation_rate

```
real(prec) tracer_plastic_mod::plastic_par_class::degradation_rate  [private]
```

degradation rate of the material

Definition at line 31 of file tracer_plastic.f90.

```
31         real(prec) :: degradation_rate
```

7.19.2.2 density

```
real(prec) tracer_plastic_mod::plastic_par_class::density  [private]
```

density of the material

Definition at line 30 of file tracer_plastic.f90.

```
30         real(prec) :: density
```

7.19.2.3 particulate

```
logical tracer_plastic_mod::plastic_par_class::particulate  [private]
```

flag to indicate if the material is a particle (false) or a collection of particles (true)

Definition at line 32 of file tracer_plastic.f90.

```
32        logical    :: particulate
```

7.19.2.4 size

```
real(prec) tracer_plastic_mod::plastic_par_class::size  [private]
```

Size (radius) of the particles (equals to the tracer radius if particulate==false)

Definition at line 33 of file tracer_plastic.f90.

```
33        real(prec) :: size
```

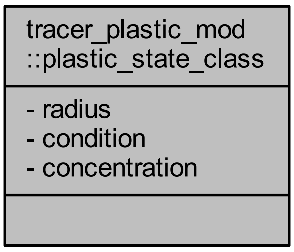
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracer_plastic.f90

7.20 tracer_plastic_mod::plastic_state_class Type Reference

Type - State variables of a tracer object representing a plastic material.

Collaboration diagram for tracer_plastic_mod::plastic_state_class:



Private Attributes

- real(prec) radius

  Tracer radius (m)
- real(prec) condition

  Material condition (1-0)
- real(prec) concentration

  Particle concentration.

7.20.1 Detailed Description

Type - State variables of a tracer object representing a plastic material.

Definition at line 36 of file tracer_plastic.f90.

7.20.2 Member Data Documentation

7.20.2.1 concentration

`real(prec) tracer_plastic_mod::plastic_state_class::concentration [private]`

Particle concentration.

Definition at line 39 of file tracer_plastic.f90.

```
39         real(prec) :: concentration
```

7.20.2.2 condition

`real(prec) tracer_plastic_mod::plastic_state_class::condition [private]`

Material condition (1-0)

Definition at line 38 of file tracer_plastic.f90.

```
38         real(prec) :: condition
```

7.20.2.3 radius

```
real(prec) tracer_plastic_mod::plastic_state_class::radius  [private]
```

Tracer radius (m)

Definition at line 37 of file tracer_plastic.f90.

```
37         real(prec) :: radius
```

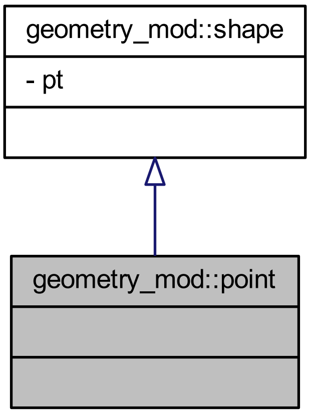
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracer_plastic.f90

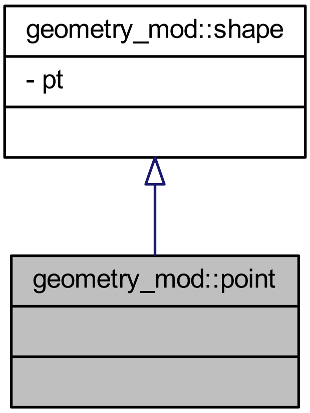
7.21 geometry_mod::point Type Reference

Type - point class.

Inheritance diagram for geometry_mod::point:



Collaboration diagram for geometry_mod::point:



7.21.1 Detailed Description

Type - point class.

Definition at line 46 of file geometry.f90.

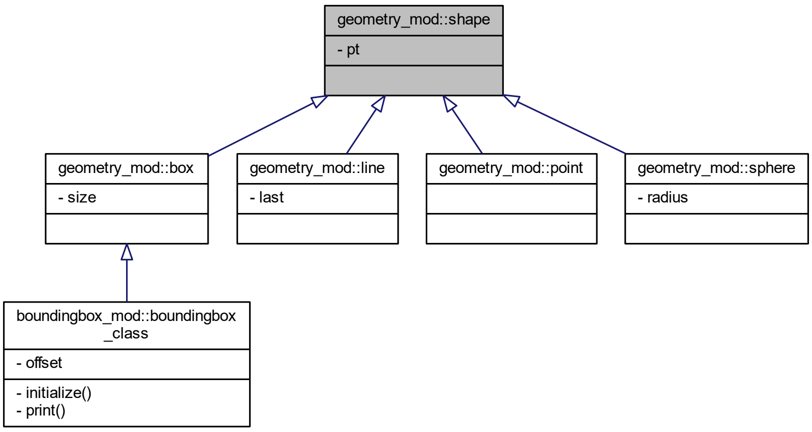
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/geometry.f90

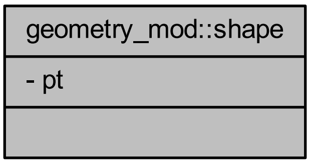
7.22 geometry_mod::shape Type Reference

Type - extendable shape class.

Inheritance diagram for geometry_mod::shape:



Collaboration diagram for geometry_mod::shape:



Private Attributes

- type(vector) pt

  Coordinates of a point.

7.22.1 Detailed Description

Type - extendable shape class.

Definition at line 42 of file geometry.f90.

7.22.2 Member Data Documentation

7.22.2.1 pt

```
type(vector) geometry_mod::shape::pt  [private]
```

Coordinates of a point.

Definition at line 43 of file geometry.f90.

```
43         type(vector) :: pt
```

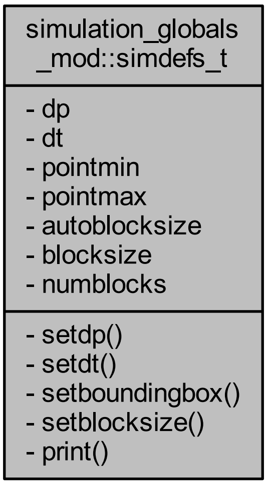
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/geometry.f90

7.23 simulation_globals_mod::simdefs_t Type Reference

Simulation definitions class.

Collaboration diagram for simulation_globals_mod::simdefs_t:



Private Member Functions

- procedure setdp
- procedure setdt
- procedure setboundingbox
- procedure setblocksize
- procedure print => printsimdefs

Private Attributes

- real(prec) dp = MV

  Initial particle spacing at source generation.
- real(prec_time) dt = MV

  Timestep for fixed step integrators (s)
- type(vector) pointmin

  Point that defines the lowest corner of the simulation bounding box.
- type(vector) pointmax

  Point that defines the upper corner of the simulation bounding box.
- logical autoblocksize = .true.

  Flag for automatic Block sizing.
- type(vector) blocksize

  Size (width & heigth) of a Block (sub-domain)
- integer numblocks

  Number of blocks in the simulation.

7.23.1 Detailed Description

Simulation definitions class.

Definition at line 45 of file simulation_globals.f90.

7.23.2 Member Function/Subroutine Documentation

7.23.2.1 print()

```
procedure simulation_globals_mod::simdefs_t::print ( )  [private]
```

Definition at line 58 of file simulation_globals.f90.

7.23.2.2 setblocksize()

```
procedure simulation_globals_mod::simdefs_t::setblocksize ( )  [private]
```

Definition at line 57 of file simulation_globals.f90.

7.23.2.3 setboundingbox()

```
procedure simulation_globals_mod::simdefs_t::setboundingbox ( )  [private]
```

Definition at line 56 of file simulation_globals.f90.

7.23.2.4 setdp()

```
procedure simulation_globals_mod::simdefs_t::setdp ( )  [private]
```

Definition at line 54 of file simulation_globals.f90.

7.23.2.5 setdt()

```
procedure simulation_globals_mod::simdefs_t::setdt ( )  [private]
```

Definition at line 55 of file simulation_globals.f90.

7.23.3 Member Data Documentation

7.23.3.1 autoblocksize

`logical simulation_globals_mod::simdefs_t::autoblocksize = .true.  [private]`

Flag for automatic Block sizing.

Definition at line 50 of file simulation_globals.f90.

```
50         logical         ::  autoblocksize = .true.
```

7.23.3.2 blocksize

`type(vector) simulation_globals_mod::simdefs_t::blocksize  [private]`

Size (width & heigth) of a Block (sub-domain)

Definition at line 51 of file simulation_globals.f90.

```
51         type(vector)    ::  blocksize
```

7.23.3.3 dp

`real(prec) simulation_globals_mod::simdefs_t::dp = MV  [private]`

Initial particle spacing at source generation.

Definition at line 46 of file simulation_globals.f90.

```
46         real(prec)      ::  Dp = mv
```

7.23.3.4 dt

`real(prec_time) simulation_globals_mod::simdefs_t::dt = MV  [private]`

Timestep for fixed step integrators (s)

Definition at line 47 of file simulation_globals.f90.

```
47         real(prec_time) ::  dt = mv
```

7.23.3.5 numblocks

```
integer simulation_globals_mod::simdefs_t::numblocks  [private]
```

Number of blocks in the simulation.

Definition at line 52 of file simulation_globals.f90.

```
52         integer         ::  numblocks
```

7.23.3.6 pointmax

```
type(vector) simulation_globals_mod::simdefs_t::pointmax  [private]
```

Point that defines the upper corner of the simulation bounding box.

Definition at line 49 of file simulation_globals.f90.

```
49         type(vector)    ::  Pointmax
```

7.23.3.7 pointmin

```
type(vector) simulation_globals_mod::simdefs_t::pointmin  [private]
```

Point that defines the lowest corner of the simulation bounding box.

Definition at line 48 of file simulation_globals.f90.

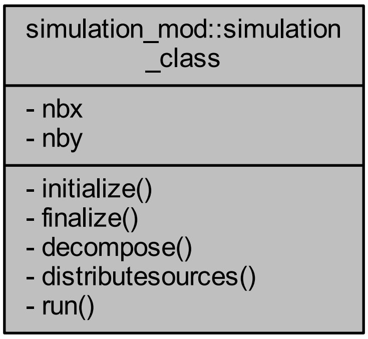
```
48         type(vector)    ::  Pointmin
```

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

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/simulation_globals.f90

7.24 simulation_mod::simulation_class Type Reference

Collaboration diagram for simulation_mod::simulation_class:



Private Member Functions

- procedure initialize => initSimulation
- procedure finalize => closeSimulation
- procedure decompose => DecomposeDomain
- procedure distributesources
- procedure run

Private Attributes

- integer nbx
- integer nby

  number of blocks in 2D

7.24.1 Detailed Description

Definition at line 33 of file simulation.f90.

7.24.2 Member Function/Subroutine Documentation

7.24.2.1 decompose()

```
procedure simulation_mod::simulation_class::decompose ( )  [private]
```

Definition at line 38 of file simulation.f90.

7.24.2.2 distributesources()

```
procedure simulation_mod::simulation_class::distributesources ( ) [private]
```

Definition at line 39 of file simulation.f90.

7.24.2.3 finalize()

```
procedure simulation_mod::simulation_class::finalize ( ) [private]
```

Definition at line 37 of file simulation.f90.

7.24.2.4 initialize()

```
procedure simulation_mod::simulation_class::initialize ( ) [private]
```

Definition at line 36 of file simulation.f90.

7.24.2.5 run()

```
procedure simulation_mod::simulation_class::run ( ) [private]
```

Definition at line 40 of file simulation.f90.

7.24.3 Member Data Documentation

7.24.3.1 nbx

```
integer simulation_mod::simulation_class::nbx [private]
```

Definition at line 34 of file simulation.f90.

```
34         integer :: nbx, nby
```

7.24.3.2 nby

```
integer simulation_mod::simulation_class::nby  [private]
```

number of blocks in 2D

Definition at line 34 of file simulation.f90.

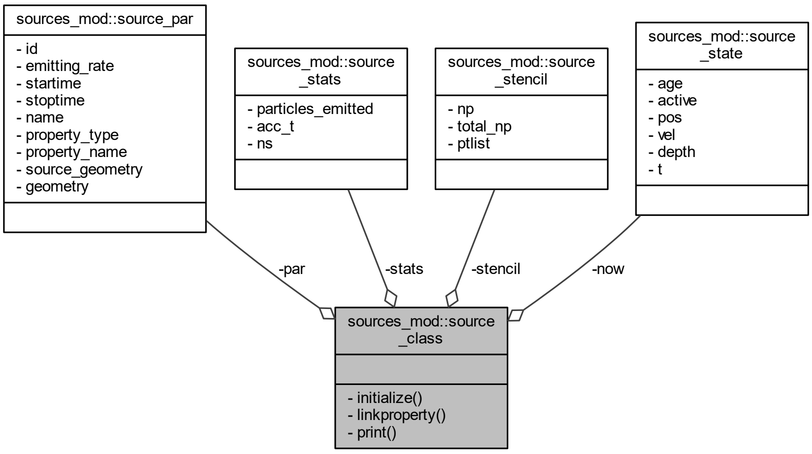
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/simulation.f90

7.25 sources_mod::source_class Type Reference

Type - The source class.

Collaboration diagram for sources_mod::source_class:



Private Member Functions

- procedure initialize => initializeSource
- procedure linkproperty
- procedure print => printSource

Private Attributes

- type(source_par) par

  To access parameters.
- type(source_state) now

  To access state variables.
- type(source_stencil) stencil

  To acess stencil variables.
- type(source_stats) stats

  To access statistics.

7.25.1 Detailed Description

Type - The source class.

Definition at line 62 of file sources.f90.

7.25.2 Member Function/Subroutine Documentation

7.25.2.1 initialize()

```
procedure sources_mod::source_class::initialize ( )  [private]
```

Definition at line 68 of file sources.f90.

7.25.2.2 linkproperty()

```
procedure sources_mod::source_class::linkproperty ( )  [private]
```

Definition at line 69 of file sources.f90.

7.25.2.3 print()

```
procedure sources_mod::source_class::print ( )  [private]
```

Definition at line 70 of file sources.f90.

7.25.3 Member Data Documentation

7.25.3.1 now

```
type(source_state) sources_mod::source_class::now  [private]
```

To access state variables.

Definition at line 64 of file sources.f90.

```
64        type(source_state) :: now
```

7.25.3.2 par

```
type(source_par) sources_mod::source_class::par [private]
```

To access parameters.

Definition at line 63 of file sources.f90.

```
63         type(source_par)   :: par
```

7.25.3.3 stats

```
type(source_stats) sources_mod::source_class::stats [private]
```

To access statistics.

Definition at line 66 of file sources.f90.

```
66         type(source_stats) :: stats
```

7.25.3.4 stencil

```
type(source_stencil) sources_mod::source_class::stencil [private]
```

To acess stencil variables.

Definition at line 65 of file sources.f90.

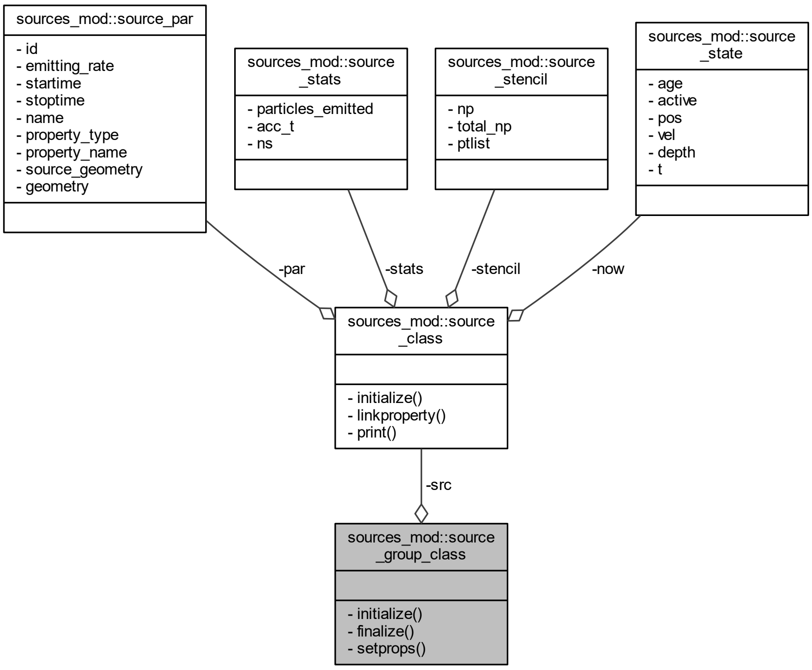
```
65         type(source_stencil) :: stencil
```

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

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/sources.f90

7.26 sources_mod::source_group_class Type Reference

Collaboration diagram for sources_mod::source_group_class:



Private Member Functions

- procedure initialize => initSources
- procedure finalize => killSources
- procedure setprops

Private Attributes

- type(source_class), dimension(:), allocatable src

7.26.1 Detailed Description

Definition at line 73 of file sources.f90.

7.26.2 Member Function/Subroutine Documentation

7.26.2.1 finalize()

```
procedure sources_mod::source_group_class::finalize ( )  [private]
```

Definition at line 77 of file sources.f90.

7.26.2.2 initialize()

```
procedure sources_mod::source_group_class::initialize ( )  [private]
```

Definition at line 76 of file sources.f90.

7.26.2.3 setprops()

```
procedure sources_mod::source_group_class::setprops ( )  [private]
```

Definition at line 78 of file sources.f90.

7.26.3 Member Data Documentation

7.26.3.1 src

```
type(source_class), dimension(:), allocatable sources_mod::source_group_class::src  [private]
```

Definition at line 74 of file sources.f90.

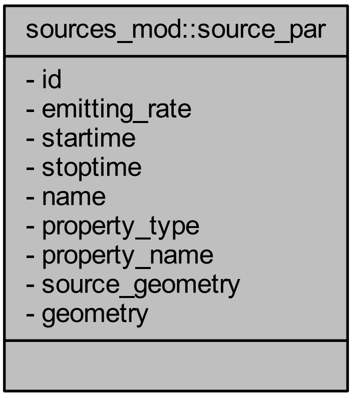
```
74         type(source_class), allocatable, dimension(:) :: src
```

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

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/sources.f90

7.27 sources_mod::source_par Type Reference

Collaboration diagram for sources_mod::source_par:



Private Attributes

- integer id

  unique source identification (integer)
- real(prec_time) emitting_rate

  Emitting rate of the source (Hz)
- real(prec_time) startime

  time to start emitting tracers
- real(prec_time) stoptime

  time to stop emitting tracers
- type(string) name

  source name
- type(string) property_type

  source property type (plastic, paper, fish, etc)
- type(string) property_name

  source property name
- type(string) source_geometry

  Source type : 'point', 'line', 'sphere', 'box'.
- class(shape), allocatable geometry

  Source geometry.

7.27.1 Detailed Description

Definition at line 27 of file sources.f90.

7.27.2 Member Data Documentation

7.27.2.1 emitting_rate

```
real(prec_time) sources_mod::source_par::emitting_rate  [private]
```

Emitting rate of the source (Hz)

Definition at line 29 of file sources.f90.

```
29        real(prec_time) :: emitting_rate
```

7.27.2.2 geometry

```
class(shape), allocatable sources_mod::source_par::geometry  [private]
```

Source geometry.

Definition at line 36 of file sources.f90.

```
36        class(shape), allocatable :: geometry
```

7.27.2.3 id

```
integer sources_mod::source_par::id  [private]
```

unique source identification (integer)

Definition at line 28 of file sources.f90.

```
28        integer :: id
```

7.27.2.4 name

```
type(string) sources_mod::source_par::name  [private]
```

source name

Definition at line 32 of file sources.f90.

```
32        type(string) :: name
```

7.27.2.5 property_name

`type(string) sources_mod::source_par::property_name [private]`

source property name

Definition at line 34 of file sources.f90.

```
34          type(string) :: property_name
```

7.27.2.6 property_type

`type(string) sources_mod::source_par::property_type [private]`

source property type (plastic, paper, fish, etc)

Definition at line 33 of file sources.f90.

```
33          type(string) :: property_type
```

7.27.2.7 source_geometry

`type(string) sources_mod::source_par::source_geometry [private]`

Source type : 'point', 'line', 'sphere', 'box'.

Definition at line 35 of file sources.f90.

```
35          type(string) :: source_geometry
```

7.27.2.8 startime

`real(prec_time) sources_mod::source_par::startime [private]`

time to start emitting tracers

Definition at line 30 of file sources.f90.

```
30          real(prec_time) :: startime
```

7.27.2.9 stoptime

```
real(prec_time) sources_mod::source_par::stoptime [private]
```

time to stop emitting tracers

Definition at line 31 of file sources.f90.

```
31         real(prec_time) :: stoptime
```

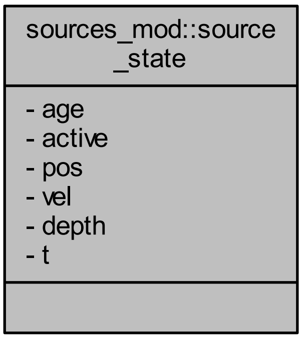
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/sources.f90

7.28 sources_mod::source_state Type Reference

Type - state variables of a source object.

Collaboration diagram for sources_mod::source_state:



Private Attributes

- real(prec_time) age
- logical active

  active switch
- type(vector) pos

  Position of the source baricenter (m)
- type(vector) vel

  Velocity of the source (m s-1)
- real(prec) depth

  Depth of the source baricenter (m)
- real(prec) t

  Temperature of the source (Celcius)

7.28.1 Detailed Description

Type - state variables of a source object.

Definition at line 39 of file sources.f90.

7.28.2 Member Data Documentation

7.28.2.1 active

```
logical sources_mod::source_state::active  [private]
```

active switch

Definition at line 41 of file sources.f90.

```
41         logical :: active
```

7.28.2.2 age

```
real(prec_time) sources_mod::source_state::age  [private]
```

Definition at line 40 of file sources.f90.

```
40         real(prec_time) :: age              ! time variables
```

7.28.2.3 depth

```
real(prec) sources_mod::source_state::depth  [private]
```

Depth of the source baricenter (m)

Definition at line 44 of file sources.f90.

```
44         real(prec) :: depth
```

7.28.2.4 pos

```
type(vector) sources_mod::source_state::pos  [private]
```

Position of the source baricenter (m)

Definition at line 42 of file sources.f90.

```
42         type(vector) :: pos
```

7.28.2.5 t

```
real(prec) sources_mod::source_state::t  [private]
```

Temperature of the source (Celcius)

Definition at line 45 of file sources.f90.

```
45         real(prec) :: T
```

7.28.2.6 vel

```
type(vector) sources_mod::source_state::vel  [private]
```

Velocity of the source (m s-1)

Definition at line 43 of file sources.f90.

```
43         type(vector) :: vel
```

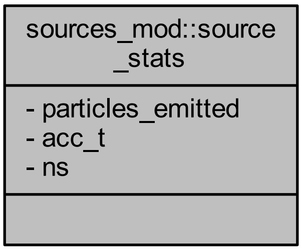
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/sources.f90

7.29 sources_mod::source_stats Type Reference

Type - statistical variables of a source object.

Collaboration diagram for sources_mod::source_stats:



Private Attributes

- integer particles_emitted

    Number of emitted particles by this source.
- real(prec_wrt) acc_t

    Accumulated temperature of the tracer (Celcius)
- integer ns

    Number of sampling steps.

7.29.1 Detailed Description

Type - statistical variables of a source object.

Definition at line 48 of file sources.f90.

7.29.2 Member Data Documentation

7.29.2.1 acc_t

```
real(prec_wrt) sources_mod::source_stats::acc_t  [private]
```

Accumulated temperature of the tracer (Celcius)

Definition at line 52 of file sources.f90.

```
52         real(prec_wrt) :: acc_T
```

7.29.2.2 ns

```
integer sources_mod::source_stats::ns  [private]
```

Number of sampling steps.

Definition at line 53 of file sources.f90.

```
53         integer :: ns
```

7.29.2.3 particles_emitted

```
integer sources_mod::source_stats::particles_emitted  [private]
```

Number of emitted particles by this source.

Definition at line 51 of file sources.f90.

```
51         integer :: particles_emitted
```

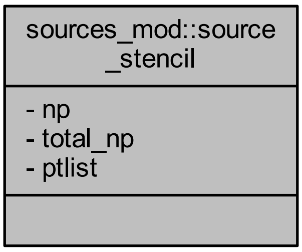
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/sources.f90

7.30 sources_mod::source_stencil Type Reference

Type - holder for the tracer creation stencil of the source.

Collaboration diagram for sources_mod::source_stencil:



Private Attributes

- integer np

  Number of tracers by emission.
- integer total_np

  Total number of tracers that this source will generate.
- type(vector), dimension(:), allocatable ptlist

  list of points (coordinates), relative to the source geometry point, to be generated at every emission

7.30.1 Detailed Description

Type - holder for the tracer creation stencil of the source.

Definition at line 56 of file sources.f90.

7.30.2 Member Data Documentation

7.30.2.1 np

```
integer sources_mod::source_stencil::np  [private]
```

Number of tracers by emission.

Definition at line 57 of file sources.f90.

```
57         integer :: np
```

7.30.2.2 ptlist

```
type(vector), dimension(:), allocatable sources_mod::source_stencil::ptlist  [private]
```

list of points (coordinates), relative to the source geometry point, to be generated at every emission

Definition at line 59 of file sources.f90.

```
59         type(vector), allocatable, dimension(:) :: ptlist
```

7.30.2.3 total_np

```
integer sources_mod::source_stencil::total_np  [private]
```

Total number of tracers that this source will generate.

Definition at line 58 of file sources.f90.

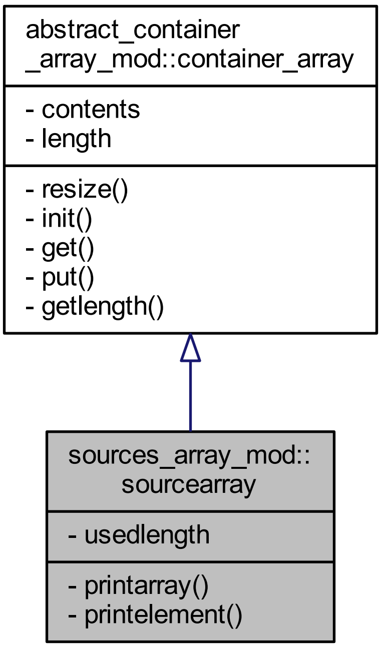
```
58        integer :: total_np
```

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

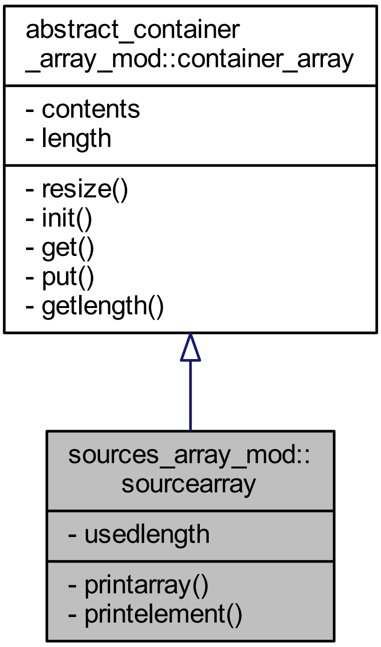
- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/sources.f90

7.31 sources_array_mod::sourcearray Type Reference

Inheritance diagram for sources_array_mod::sourcearray:



Collaboration diagram for sources_array_mod::sourcearray:



Private Member Functions

- procedure printarray => print_SourceArray
- procedure printelement => print_SourceArray_Element

Private Attributes

- integer usedlength

7.31.1 Detailed Description

Definition at line 26 of file sources_array.f90.

7.31.2 Member Function/Subroutine Documentation

7.31.2.1 printarray()

```
procedure sources_array_mod::sourcearray::printarray ( )  [private]
```

Definition at line 29 of file sources_array.f90.

7.31.2.2 printelement()

```
procedure sources_array_mod::sourcearray::printelement ( )  [private]
```

Definition at line 30 of file sources_array.f90.

7.31.3 Member Data Documentation

7.31.3.1 usedlength

```
integer sources_array_mod::sourcearray::usedlength  [private]
```

Definition at line 27 of file sources_array.f90.

```
27         integer :: usedLength
```

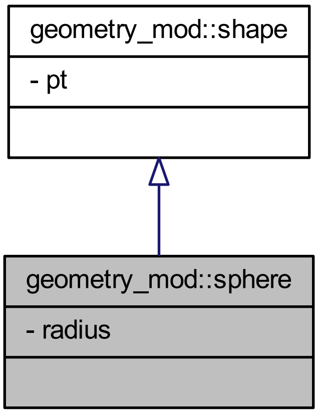
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/sources_array.f90

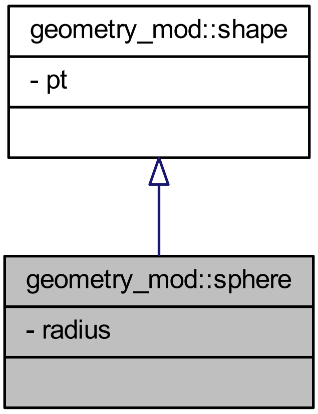
7.32 geometry_mod::sphere Type Reference

Type - sphere class.

Inheritance diagram for geometry_mod::sphere:



Collaboration diagram for geometry_mod::sphere:



Private Attributes

- real(prec) radius

  Sphere radius.

7.32.1 Detailed Description

Type - sphere class.

Definition at line 53 of file geometry.f90.

7.32.2 Member Data Documentation

7.32.2.1 radius

```
real(prec) geometry_mod::sphere::radius  [private]
```

Sphere radius.

Definition at line 54 of file geometry.f90.

```
54         real(prec) :: radius
```

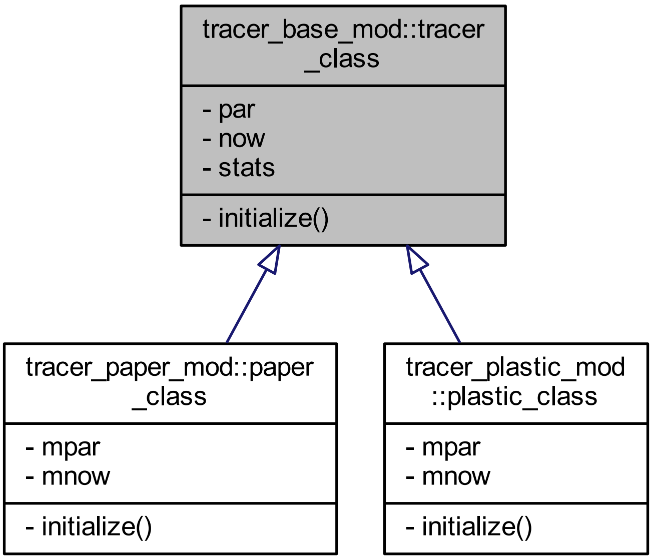
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/geometry.f90

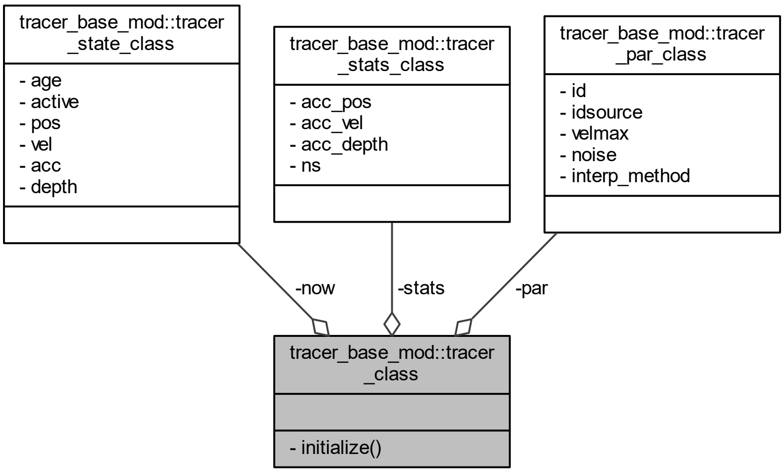
7.33 tracer_base_mod::tracer_class Type Reference

Type - The pure Lagrangian tracer class.

Inheritance diagram for tracer_base_mod::tracer_class:


tracer_base_mod::tracer_class
- par
- now
- stats
- initialize()
tracer_paper_mod::paper_class
- mpar
- mnow
- initialize()
tracer_plastic_mod::plastic_class
- mpar
- mnow
- initialize()

Collaboration diagram for tracer_base_mod::tracer_class:



Private Member Functions

- procedure initialize

Private Attributes

- type(tracer_par_class) par

  To access parameters.
- type(tracer_state_class) now

  To access state variables.
- type(tracer_stats_class) stats

  To access statistics.

7.33.1 Detailed Description

Type - The pure Lagrangian tracer class.

Definition at line 55 of file tracer_base.f90.

7.33.2 Member Function/Subroutine Documentation

7.33.2.1 initialize()

```
procedure tracer_base_mod::tracer_class::initialize ( )  [private]
```

Definition at line 60 of file tracer_base.f90.

7.33.3 Member Data Documentation

7.33.3.1 now

```
type(tracer_state_class) tracer_base_mod::tracer_class::now  [private]
```

To access state variables.

Definition at line 57 of file tracer_base.f90.

```
57        type(tracer_state_class) :: now
```

7.33.3.2 par

```
type(tracer_par_class) tracer_base_mod::tracer_class::par  [private]
```

To access parameters.

Definition at line 56 of file tracer_base.f90.

```
56         type(tracer_par_class)   :: par
```

7.33.3.3 stats

```
type(tracer_stats_class) tracer_base_mod::tracer_class::stats  [private]
```

To access statistics.

Definition at line 58 of file tracer_base.f90.

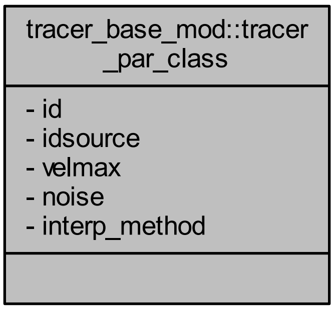
```
58         type(tracer_stats_class) :: stats
```

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

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracer_base.f90

7.34 tracer_base_mod::tracer_par_class Type Reference

Collaboration diagram for tracer_base_mod::tracer_par_class:



Private Attributes

- integer id

  unique tracer identification
- integer idsource

  Source to which the tracer belongs.
- real(prec) velmax

  Maximum velocity of tracer to track (m/s)
- logical noise
- type(string) interp_method

  interpolation method this tracer calls

7.34.1 Detailed Description

Definition at line 27 of file tracer_base.f90.

7.34.2 Member Data Documentation

7.34.2.1 id

```
integer tracer_base_mod::tracer_par_class::id  [private]
```

unique tracer identification

Definition at line 28 of file tracer_base.f90.

```
28         integer :: id
```

7.34.2.2 idsource

```
integer tracer_base_mod::tracer_par_class::idsource  [private]
```

Source to which the tracer belongs.

Definition at line 29 of file tracer_base.f90.

```
29         integer :: idsource
```

7.34.2.3 interp_method

```
type(string) tracer_base_mod::tracer_par_class::interp_method [private]
```

interpolation method this tracer calls

Definition at line 32 of file tracer_base.f90.

```
32         type(string) :: interp_method
```

7.34.2.4 noise

```
logical tracer_base_mod::tracer_par_class::noise [private]
```

Definition at line 31 of file tracer_base.f90.

```
31         logical    :: noise                !  Add noise to location
```

7.34.2.5 velmax

```
real(prec) tracer_base_mod::tracer_par_class::velmax [private]
```

Maximum velocity of tracer to track (m/s)

Definition at line 30 of file tracer_base.f90.

```
30         real(prec) :: velmax
```

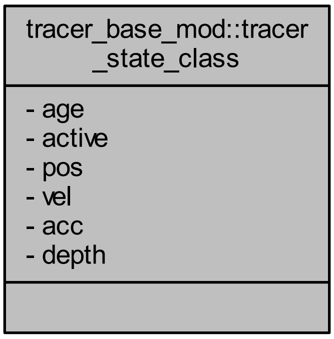
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracer_base.f90

7.35 tracer_base_mod::tracer_state_class Type Reference

Type - state variables of a pure Lagrangian tracer object.

Collaboration diagram for tracer_base_mod::tracer_state_class:



Private Attributes

- real(prec_time) age
- logical active

  active switch
- type(vector) pos

  Position of the tracer (m)
- type(vector) vel

  Velocity of the tracer (m s-1)
- type(vector) acc

  Acceleration of the tracer (m s-2)
- real(prec) depth

  Depth of the tracer (m)

7.35.1 Detailed Description

Type - state variables of a pure Lagrangian tracer object.

Definition at line 35 of file tracer_base.f90.

7.35.2 Member Data Documentation

7.35.2.1 acc

```
type(vector) tracer_base_mod::tracer_state_class::acc  [private]
```

Acceleration of the tracer (m s-2)

Definition at line 40 of file tracer_base.f90.

```
40        type(vector) :: acc
```

7.35.2.2 active

```
logical tracer_base_mod::tracer_state_class::active  [private]
```

active switch

Definition at line 37 of file tracer_base.f90.

```
37        logical :: active
```

7.35.2.3 age

```
real(prec_time) tracer_base_mod::tracer_state_class::age  [private]
```

Definition at line 36 of file tracer_base.f90.

```
36         real(prec_time) :: age             ! time variables
```

7.35.2.4 depth

```
real(prec) tracer_base_mod::tracer_state_class::depth  [private]
```

Depth of the tracer (m)

Definition at line 41 of file tracer_base.f90.

```
41         real(prec) :: depth
```

7.35.2.5 pos

```
type(vector) tracer_base_mod::tracer_state_class::pos  [private]
```

Position of the tracer (m)

Definition at line 38 of file tracer_base.f90.

```
38         type(vector) :: pos
```

7.35.2.6 vel

```
type(vector) tracer_base_mod::tracer_state_class::vel  [private]
```

Velocity of the tracer (m s-1)

Definition at line 39 of file tracer_base.f90.

```
39         type(vector) :: vel
```

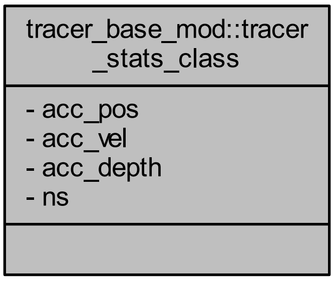
The documentation for this type was generated from the following file:

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracer_base.f90

7.36 tracer_base_mod::tracer_stats_class Type Reference

Type - statistical variables of a pure Lagrangian tracer object.

Collaboration diagram for tracer_base_mod::tracer_stats_class:



Private Attributes

- type(vector) acc_pos

  Accumulated position of the tracer (m)
- type(vector) acc_vel

  Accumulated velocity of the tracer (m s-1)
- real(prec_wrt) acc_depth

  Accumulated depth of the tracer (m)
- integer ns

  Number of sampling steps.

7.36.1 Detailed Description

Type - statistical variables of a pure Lagrangian tracer object.

Definition at line 45 of file tracer_base.f90.

7.36.2 Member Data Documentation

7.36.2.1 acc_depth

```
real(prec_wrt) tracer_base_mod::tracer_stats_class::acc_depth  [private]
```

Accumulated depth of the tracer (m)

Definition at line 50 of file tracer_base.f90.

```
50         real(prec_wrt) :: acc_depth
```

7.36.2.2 acc_pos

```
type(vector) tracer_base_mod::tracer_stats_class::acc_pos  [private]
```

Accumulated position of the tracer (m)

Definition at line 48 of file tracer_base.f90.

```
48         type(vector) :: acc_pos
```

7.36.2.3 acc_vel

```
type(vector) tracer_base_mod::tracer_stats_class::acc_vel  [private]
```

Accumulated velocity of the tracer (m s-1)

Definition at line 49 of file tracer_base.f90.

```
49         type(vector) :: acc_vel
```

7.36.2.4 ns

```
integer tracer_base_mod::tracer_stats_class::ns  [private]
```

Number of sampling steps.

Definition at line 52 of file tracer_base.f90.

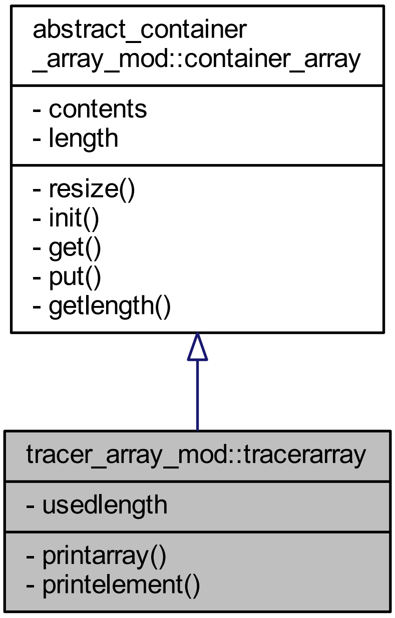
```
52         integer :: ns
```

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

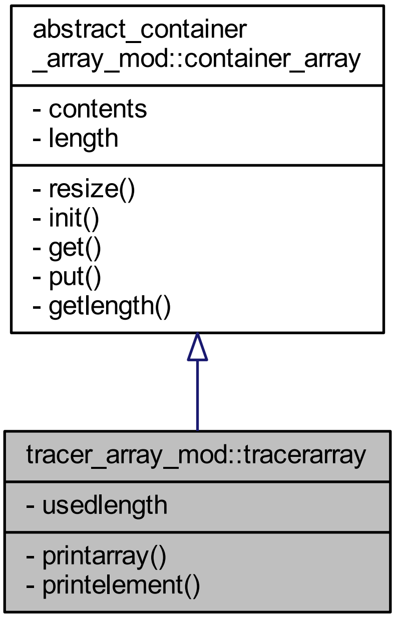
- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracer_base.f90

7.37 tracer_array_mod::tracerarray Type Reference

Inheritance diagram for tracer_array_mod::tracerarray:



Collaboration diagram for tracer_array_mod::tracerarray:



Private Member Functions

- procedure printarray => print_TracerArray
- procedure printelement => print_TracerArray_Element

Private Attributes

- integer usedlength

7.37.1 Detailed Description

Definition at line 26 of file tracer_array.f90.

7.37.2 Member Function/Subroutine Documentation

7.37.2.1 printarray()

```
procedure tracer_array_mod::tracerarray::printarray ( )  [private]
```

Definition at line 29 of file tracer_array.f90.

7.37.2.2 printelement()

```
procedure tracer_array_mod::tracerarray::printelement ( )  [private]
```

Definition at line 30 of file tracer_array.f90.

7.37.3 Member Data Documentation

7.37.3.1 usedlength

```
integer tracer_array_mod::tracerarray::usedlength  [private]
```

Definition at line 27 of file tracer_array.f90.

```
27         integer :: usedLength
```

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

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracer_array.f90

Chapter 8

File Documentation

8.1 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/README.md File Reference

8.2 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/app/MOHID↩ Lagrangian.f90 File Reference

Functions/Subroutines

- program mohidlagrangian

8.2.1 Function/Subroutine Documentation

8.2.1.1 mohidlagrangian()

```
program mohidlagrangian ( )
```

Definition at line 17 of file MOHIDLagrangian.f90.

8.3 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/about.f90 File Reference

Modules

- module about_mod

  Module to print version, licence, preambles.

Functions/Subroutines

- subroutine, public about_mod::printlicpreamble

  Public licence and preamble printer routine.

Variables

- type(string) about_mod::version
- type(string) about_mod::author
- type(string) about_mod::date

8.4 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/abstract_↩ container_array.f90 File Reference

Data Types

- type abstract_container_array_mod::container_array

Modules

- module abstract_container_array_mod

  Module that defines an unlimited polymorphic container class and related methods. A container is a fundamental entity allowing to build data structures such as lists and arrays. This is an abstract type, so a derived type must be defined for any specific contents that may be required. Those derived types should provide type-specific methods that require type-guards, such as printing.

Functions/Subroutines

- class(∗) function, pointer abstract_container_array_mod::getvalue (this, index)

  Method that returns returns the requested entry (pointer)
- subroutine abstract_container_array_mod::putvalue (this, index, value)

  Method that stores a value on the requested index.
- integer function abstract_container_array_mod::getlength (this)

  Method that returns the length of the array.
- subroutine abstract_container_array_mod::resizearray (this, newsize)

  Method that grows (adds empty space) or shrinks (discards the last entries) of the array. Use sparsely as this might get expensive for large array operations. Should think of a way to use move_alloc()
- subroutine abstract_container_array_mod::initarray (this, entries, tocopy)

  Method that allocates the container array. Deallocates if already allocated.

8.5 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/blocks.f90 File Reference

Data Types

- type blocks_mod::block_class

Modules

- module blocks_mod

  Module that defines a block class and related methods. A block is a fundamental type of the model. It contains a sub-domain of the simulation bounding box, holding all entities inside that sub-domain. It maps to a domain decomposition parallelization strategy, if needed.

Functions/Subroutines

- subroutine blocks_mod::initblock (self, id, templatebox)

  method to allocate and initialize blocks and their emitters
- subroutine blocks_mod::putsource (self, sourcetoput)

  Method to place a Source on the Block SourceArray. Checks for space and allocates more if needed. The array gets incremented by une unit at a time.
- subroutine blocks_mod::printblock (self)

  Method to print basic info about the block.
- subroutine blocks_mod::printdetailblock (self)

  Method to print detailed info about the block.
- subroutine, public blocks_mod::setblocks (auto, nblk, nxi, nyi)

  routine to set the simulation blocks extents and call the block initializer
- subroutine, public blocks_mod::allocblocks (nblk)

  routine to allocate the simulation blocks

Variables

- type(block_class), dimension(:), allocatable, public blocks_mod::dblock

8.6 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/boundingbox.f90 File Reference

Data Types

- type boundingbox_mod::boundingbox_class

Modules

- module boundingbox_mod

  Module that defines a simulation Bounding Box.

Functions/Subroutines

- subroutine boundingbox_mod::initboundingbox (self)

  Method to initialize the simulation Bounding Box.
- subroutine boundingbox_mod::printboundingbox (self)

  Method to print the simulation Bounding Box.

Variables

- type(boundingbox_class), public boundingbox_mod::bbox

8.7 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/common_↩ modules.f90 File Reference

Modules

- module commom_modules

  Module to hold all of the commonly used base modules.

8.8 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/container.f90 File Reference

Data Types

- interface container_mod::container
- interface container_mod::container

Modules

- module container_mod

  Module that defines an unlimited polymorphic container class and related methods. A container is a fundamental entity allowing to build data structures such as lists and arrays.

Functions/Subroutines

- class(∗) function, pointer container_mod::getcontent (this)

  Method that returns a pointer to the values stored in the container.
- subroutine container_mod::storecontent (this, to_store)

  Method that stores the provided value in the container using sourced allocation.
- subroutine container_mod::printcontainer (this)

  Method to print the stored value. Only knows about instrinsic types, ignores (but warns) if other types are passed.
- class(container) function, pointer container_mod::constructor (to_store)

  Container constructor, can be used with the 'container' name since it is defined as an interface.

8.9 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/emitter.f90 File Reference

Data Types

- type emitter_mod::emitter_class

Modules

- module emitter_mod

  Module that defines an emitter class and related methods. This module is responsible for building a potential tracer list based on the availble sources and calling their initializers.

Functions/Subroutines

- subroutine emitter_mod::initracers (self, srcs)

  method that calls the tracer initialization from the emmiter object
- subroutine emitter_mod::alloctracers (self, src)

  method that allocates the tracers respective to a given source
- subroutine emitter_mod::initializeemitter (self)

  method that initializes an emmiter class object. Sets default values
- subroutine emitter_mod::addsource (self, src)

  method to compute the total emittable particles per source and allocate them
- subroutine emitter_mod::setotalnp (src)

  private routine that returns the total number of tracers an input source will potentially create

8.10 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/geometry.f90 File Reference

Data Types

- type geometry_mod::geometry_class
- type geometry_mod::shape

  Type - extendable shape class.
- type geometry_mod::point

  Type - point class.
- type geometry_mod::line

  Type - line class.
- type geometry_mod::sphere

  Type - sphere class.
- type geometry_mod::box

  Type - point class.

Modules

- module geometry_mod

  Module that defines geometry classes and related methods.

Functions/Subroutines

- subroutine geometry_mod::allocatelist (self)

  Public routine to allocate the possible geometry name list.
- logical function geometry_mod::inlist (self, geomname)

  Public function that returns a logical if the input geometry name is valid.
- integer function geometry_mod::fillsize (self, shapetype)

  method to get the number of points that fill a given geometry
- subroutine geometry_mod::fill (self, shapetype, fillsize, ptlist)

  method to get the list of points that fill a given geometry
- subroutine geometry_mod::printgeometry (self, shapetype)

  method to print the details of a given geometry
- integer function geometry_mod::sphere_np_count (dp, r)

  private function that returns the number of points distributed on a grid with spacing dp inside a sphere
- subroutine geometry_mod::sphere_grid (dp, r, np, ptlist)

  private routine that returns the points distributed on a grid with spacing dp inside a sphere
- subroutine geometry_mod::box_grid (dp, size, np, ptlist)

  private routine that returns the points distributed on a grid with spacing dp inside a box
- subroutine geometry_mod::line_grid (dp, dist, np, ptlist)

  private routine that returns the points distributed on a grid with spacing dp along a line

Variables

- type(geometry_class), public geometry_mod::geometry

8.11 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/initialize.f90 File Reference

Modules

- module initialize_mod

  Module with the simulation initialization related definitions and methods. Has one public access routine that is in-charge of building the simulation space from input files.

Functions/Subroutines

- subroutine initialize_mod::linkpropertysources (linksNode)

  Private property xml parser routine. Reads the properties tab from the xml file and links these to the corresponding source.
- subroutine initialize_mod::init_properties (case_node)

  Private property xml parser routine. Reads the properties tab from the xml file and links these to the corresponding source.
- subroutine initialize_mod::read_xml_geometry (source, source_detail, source_shape)

  Private geometry xml parser routine. Reads a geometry from the xml depending on the geometry type of the node.
- subroutine initialize_mod::init_sources (case_node)

  Private source definitions parser routine. Builds the tracer sources from the input xml case file.
- subroutine initialize_mod::init_simdefs (case_node)

  Private simulation definitions parser routine. Builds the simulation geometric space from the input xml case file.
- subroutine initialize_mod::init_caseconstants (case_node)

  Private case constant parser routine. Builds the simulation parametric space from the input xml case file.
- subroutine initialize_mod::init_parameters (execution_node)

  Private parameter parser routine. Builds the simulation parametric space from the input xml case file.
- subroutine, public initialize_mod::initfromxml (xmlfilename)

  Public xml parser routine. Builds the simulation space from the input xml case file.

8.12 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/simulation.f90 File Reference

Data Types

- type simulation_mod::simulation_class

Modules

- module simulation_mod

  Module to hold the simulation class and its methods.

Functions/Subroutines

- subroutine simulation_mod::run (self)

  Simulation run method. Runs the initialized case main time cycle.
- subroutine simulation_mod::initsimulation (self, casefilename, outpath)

  Simulation initialization method. Effectively builds and populates the simulation objects that will be used latter on.
- subroutine simulation_mod::distributesources (self)

  Simulation to distribute the Sources to the blocks.
- subroutine simulation_mod::decomposedomain (self)

  Simulation method to do domain decomposition and define the Blocks.
- subroutine simulation_mod::closesimulation (self)

  Simulation finishing method. Closes output files and writes the final messages.

8.13 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/simulation_↩ globals.f90 File Reference

Data Types

- type simulation_globals_mod::parameters_t
- type simulation_globals_mod::simdefs_t

  Simulation definitions class.
- type simulation_globals_mod::constants_t

  Case Constants class.
- type simulation_globals_mod::filenames_t

  File names class.
- type simulation_globals_mod::globals_class

  Globals class - This is a container for every global variable on the simulation.

Modules

- module simulation_globals_mod

  Module to hold the simulation global parameter classes and their methods.

Functions/Subroutines

- subroutine simulation_globals_mod::setdefaults (self)

  Globals default setting routine.
- subroutine simulation_globals_mod::setparameter (self, parmkey, parmvalue)

  Private parameter setting method. Builds the simulation parametric space from the input case file.
- subroutine simulation_globals_mod::check (self)

  Parameter checking method. Checks if mandatory parameters were set.
- subroutine simulation_globals_mod::printsimparameters (self)

  Parameter printing method.
- subroutine simulation_globals_mod::getintegratorname (name, code)

  Routine to get integrator scheme name.
- subroutine simulation_globals_mod::setgravity (self, grav)

  Gravity setting routine.
- subroutine simulation_globals_mod::setz0 (self, read_z0)

  Z0 setting routine.
- subroutine simulation_globals_mod::setrho (self, read_rho)

  Rho_Ref setting routine.
- subroutine simulation_globals_mod::printconstants (self)

  Public constants printing routine.
- subroutine simulation_globals_mod::setdp (self, read_dp)

  Dp setting routine.
- subroutine simulation_globals_mod::setdt (self, read_dt)

  Dt setting routine.
- subroutine simulation_globals_mod::setboundingbox (self, point_, coords)

  Bounding box setting routine.
- subroutine simulation_globals_mod::setblocksize (self, bsize)

  blocksize box setting routine.
- subroutine simulation_globals_mod::printsimdefs (self)

  Public simulation definitions printing routine.

Variables

- type(globals_class), public simulation_globals_mod::globals

8.14 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/simulation_logger.f90 File Reference

Data Types

- type simulation_logger_mod::logger_class

Modules

- module simulation_logger_mod

  Module to hold all the simulation logger related definitions and methods.

Functions/Subroutines

- subroutine simulation_logger_mod::initlog (self, outpath)

  Log file initizalization routine.
- subroutine simulation_logger_mod::closelog (self)

  Log file closure routine.
- subroutine simulation_logger_mod::put_inlog (self, tologstr, timeoption)

  Log serialization routine.
- subroutine, public simulation_logger_mod::gettimestamp (timestamp)

  Public timestamp builder.

Variables

- type(logger_class), public simulation_logger_mod::log

8.15 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/simulation_memory.f90 File Reference

Data Types

- type simulation_memory_mod::memory_t

Modules

- module simulation_memory_mod

  Module to hold the simulation memory managment class and its methods.

Functions/Subroutines

- subroutine simulation_memory_mod::initializememory (self)

  Private memory logger initialization method.
- subroutine simulation_memory_mod::getotal (self, size)

  Private method to retreive the total size of the allocated memory.
- subroutine simulation_memory_mod::addblock (self, size)

  Private method to add the size of a Block to the memory log.
- subroutine simulation_memory_mod::addsource (self, size)

  Private method to add the size of a Source to the memory log.
- subroutine simulation_memory_mod::addtracer (self, size)

  Private method to add the size of a Tracer to the memory log.
- subroutine simulation_memory_mod::removetracer (self, size)

  Private method to remove the size of a Tracer from the memory log.
- subroutine simulation_memory_mod::adddef (self, size)

  Private method to add the size of a definition to the memory log.
- subroutine simulation_memory_mod::printmemory (self)

  Method to print the total allocated memory.
- subroutine simulation_memory_mod::printmemorydetailed (self)

  Private method to print the allocated memory.

Variables

- type(memory_t), public simulation_memory_mod::simmemory

8.16 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/simulation_↩ precision.f90 File Reference

Modules

- module simulation_precision_mod

    Module to control the precision of the variables trough the project.

Variables

- integer, parameter simulation_precision_mod::sp = kind(1._R4P)

    Simple precision definition switch.
- integer, parameter simulation_precision_mod::dp = kind(1._R8P)

    Double precision definition switch.
- integer, parameter, public simulation_precision_mod::prec = sp
- integer, parameter, public simulation_precision_mod::prec_time = sp
- integer, parameter, public simulation_precision_mod::prec_wrt = sp
- real(prec), parameter, public simulation_precision_mod::missing_value_default = -9999.0_dp
- real(prec), parameter, public simulation_precision_mod::mv = MISSING_VALUE_DEFAULT
- real(prec), parameter, public simulation_precision_mod::mv_int = int(MISSING_VALUE_DEFAULT)
- real(prec), parameter, public simulation_precision_mod::err_dist = 1E8_dp
- integer, parameter, public simulation_precision_mod::err_ind = -1
- integer, parameter, public simulation_precision_mod::char_len = 99

8.17 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/simulation_↩ xmlparser.f90 File Reference

Modules

- module simulation_xmlparser_mod

    Module with the simulation xml parsing related definitions and routines.

Functions/Subroutines

- subroutine, public simulation_xmlparser_mod::readxmlatt (xmlnode, tag, att_name, att_value, read_flag, mandatory)

    Private attribute xml parser routine. In the format <Tag att_name="att_value".
- subroutine, public simulation_xmlparser_mod::readxmlvector (xmlnode, tag, vec, read_flag, mandatory)

    Private vector xml parser routine. Vector must be in format <Tag x="vec%x" y="vec%y" z="vec%z">
- subroutine, public simulation_xmlparser_mod::gotochildnode (currentNode, targetNode, targetNodeName, read_flag, mandatory)

    Private routine to retrieve a node within a node. Returns a nullifyed pointer if not found, stops if mandatory.

8.18 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/sources.f90 File Reference

Data Types

- type sources_mod::source_par
- type sources_mod::source_state

  Type - state variables of a source object.
- type sources_mod::source_stats

  Type - statistical variables of a source object.
- type sources_mod::source_stencil

  Type - holder for the tracer creation stencil of the source.
- type sources_mod::source_class

  Type - The source class.
- type sources_mod::source_group_class

Modules

- module sources_mod

  Module that defines a source class and related methods.

Functions/Subroutines

- subroutine sources_mod::initsources (self, nsources)

  source allocation routine - allocates sources objects
- subroutine sources_mod::killsources (self)

  source group destructor - deallocates sources objects
- subroutine sources_mod::setprops (self, srcid_str, ptype, pname)

  source property setting routine, calls source by id to set its properties
- subroutine sources_mod::initializesource (src, id, name, emitting_rate, start, finish, source_geometry, shapetype)

  source inititialization proceadure - initializes Source variables
- subroutine sources_mod::linkproperty (src, ptype, pname)

  source property setting proceadure - initializes Source variables
- subroutine sources_mod::printsource (src)

  source print routine - prints a source info on console/log

Variables

- type(source_group_class), public sources_mod::tempsources

8.19 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/sources_↩ array.f90 File Reference

Data Types

- type sources_array_mod::sourcearray

Modules

- module sources_array_mod

Functions/Subroutines

- subroutine sources_array_mod::print_sourcearray (this)
- subroutine sources_array_mod::print_sourcearray_element (this, index)

8.20 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracer_↩ array.f90 File Reference

Data Types

- type tracer_array_mod::tracerarray

Modules

- module tracer_array_mod

Functions/Subroutines

- subroutine tracer_array_mod::print_tracerarray (this)
- subroutine tracer_array_mod::print_tracerarray_element (this, index)

8.21 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracer_↩ base.f90 File Reference

Data Types

- type tracer_base_mod::tracer_par_class
- type tracer_base_mod::tracer_state_class

  Type - state variables of a pure Lagrangian tracer object.
- type tracer_base_mod::tracer_stats_class

  Type - statistical variables of a pure Lagrangian tracer object.
- type tracer_base_mod::tracer_class

  Type - The pure Lagrangian tracer class.

Modules

- module tracer_base_mod

  Module that defines a pure Lagrangian tracer class and related methods.

Functions/Subroutines

- subroutine tracer_base_mod::initialize (trc, id, id_source, time, pt)

  Tracer initialization method.

Variables

- type(tracer_class), dimension(:), allocatable, public tracer_base_mod::tracer

8.22 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracer_interp.f90 File Reference

Modules

- module tracer_interp_mod

8.23 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracer_paper.f90 File Reference

Data Types

- type tracer_paper_mod::paper_par_class
- type tracer_paper_mod::paper_state_class

  Type - State variables of a tracer object representing a paper material.
- type tracer_paper_mod::paper_class

  Type - The plastic material Lagrangian tracer class.

Modules

- module tracer_paper_mod

  Module that defines a Lagrangian tracer class for paper modelling and related methods. The type is defined as a derived type from the pule Lagrangian tracer, and hence inherits all of it's data and methods.

Functions/Subroutines

- subroutine tracer_paper_mod::paper_initialize (trc, id, id_source, time, pt)

  Tracer initialization method.

8.24 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracer_plastic.f90 File Reference

Data Types

- type tracer_plastic_mod::plastic_par_class
- type tracer_plastic_mod::plastic_state_class

  Type - State variables of a tracer object representing a plastic material.
- type tracer_plastic_mod::plastic_class

  Type - The plastic material Lagrangian tracer class.

Modules

- module tracer_plastic_mod

  Module that defines a Lagrangian tracer class for plastic modelling and related methods. The type is defined as a derived type from the pule Lagrangian tracer, and hence inherits all of it's data and methods.

Functions/Subroutines

- subroutine tracer_plastic_mod::plastic_initialize (trc, id, id_source, time, pt)

  Tracer initialization method.

8.25 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracers.f90 File Reference

Modules

- module tracers_mod

  Module to hold and wrap all the tracer respective modules. Defines a pure Lagrangian tracer block. This is intended to serve as the base class for every type of tracer class needed, that should be built as derived of this class, with the necessary modifiers to model the desired behaviour. Basic tracer data (parameters, variables) are implemented. Tracer methods such as I/O, integration and interpolation routines are implemented.

8.26 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/utilities.f90 File Reference

Modules

- module utilities_mod

  Module that provides useful back-end routines.

Functions/Subroutines

- real(prec) function, public utilities_mod::get_closest_twopow (num)

  Public function that returns the closest power of 2 or a given real number.