MOHID Lagrangian

0.2

Generated by Doxygen 1.8.14

Contents

Chapter 1

MOHIDLagrangian - Heavy development phase - Work in progress!

Check out our `code documentation page`!

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 abstract_linkedlist_mod Module Reference

Module that defines an unlimited polymorphic container list 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 linkedlist

Functions/Subroutines

- subroutine addvalue (this, value, key)

  Method that stores a value on a new link.
- subroutine removecurrent (this)

  Method that removes a link from the list.
- subroutine remove (this, n)

  Method that removes the nth link from a list.
- class(link) function, pointer getfirst (this)

  Method that returns the first link of the list.
- class(link) function, pointer getlast (this)

  Method that returns the last link of the list.
- pure integer function getsize (this)

  Method that returns the size (number of links) of a list.
- class(∗) function, pointer getvalue (this, n)

  Method that returns the value of the nth link of a list.
- class(∗) function, pointer currentvalue (this)

  Method that returns the value of the current link.
- subroutine next (this)

  Method that returns the next link in the list.
- subroutine previous (this)

  Method that returns the previous link in the list.
- pure logical function morevalues (this)

  Method that returns a logical with signaling if the current link is ok.
- subroutine reset (this)

  Method that resets the list iterator.

6.1.1 Detailed Description

Module that defines an unlimited polymorphic container list 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

6.1.2 Function/Subroutine Documentation

6.1.2.1 addvalue()

```
subroutine abstract_linkedlist_mod::addvalue (
            class(linkedlist) this,
            class(*), intent(in) value,
            integer, intent(in), optional key )  [private]
```

Method that stores a value on a new link.

Author

Ricardo Birjukovs Canelas - MARETEC

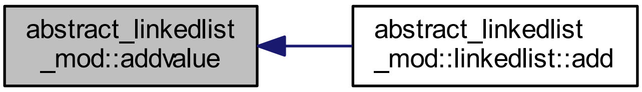
Parameters

[this,value,key]	

Definition at line 75 of file abstract_LinkedList.f90.

```
75      class(linkedlist) :: this
76      class(*), intent(in) :: value
77      integer, intent(in), optional :: key
78      class(link), pointer :: newLink
79      if (.not. associated(this%firstLink)) then
80          if (present(key)) then
81              this%firstLink => link(value, this%firstLink, this%firstLink, key)
82          else
83              this%firstLink => link(value, this%firstLink, this%firstLink)
84          end if
85          this%lastLink => this%firstLink
86      else
87          if (present(key)) then
88              newlink => link(value, this%lastLink, this%lastLink%nextLink(), key)
89          else
90              newlink => link(value, this%lastLink, this%lastLink%nextLink())
91          end if
92          call this%lastLink%setNextLink(newlink)
93          this%lastLink => newlink
94      end if
95      this%numLinks = this%numLinks + 1
```

Here is the caller graph for this function:



6.1.2.2 currentvalue()

```
class(*) function, pointer abstract_linkedlist_mod::currentvalue (
              class(linkedlist) this )  [private]
```

Method that returns the value of the current link.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 223 of file abstract_LinkedList.f90.

```
223     class(linkedlist) :: this
224     class(*), pointer :: currentValue
225     currentvalue => this%currLink%get()
```

6.1.2.3 getfirst()

```
class(link) function, pointer abstract_linkedlist_mod::getfirst (
              class(linkedlist) this )  [private]
```

Method that returns the first link of the list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 160 of file abstract_LinkedList.f90.

```
160     class(linkedlist) :: this
161     class(link), pointer :: firstlink
162     firstlink => this%firstLink
```

6.1.2.4 getlast()

```
class(link) function, pointer abstract_linkedlist_mod::getlast (
            class(linkedlist) this )  [private]
```

Method that returns the last link of the list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 171 of file abstract_LinkedList.f90.

```
171     class(linkedlist) :: this
172     class(link), pointer :: lastLink
173     lastlink => this%lastLink
```

6.1.2.5 getsize()

```
pure integer function abstract_linkedlist_mod::getsize (
            class(linkedlist), intent(in) this )  [private]
```

Method that returns the size (number of links) of a list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 182 of file abstract_LinkedList.f90.

```
182     class(linkedlist), intent(in) :: this
183     getsize = this%numLinks
```

6.1.2.6 getvalue()

```
class(*) function, pointer abstract_linkedlist_mod::getvalue (
            class(linkedlist), intent(in) this,
            integer, intent(in) n )  [private]
```

Method that returns the value of the nth link of a list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 192 of file abstract_LinkedList.f90.

```
192     class(linkedlist), intent(in) :: this
193     integer, intent(in) :: n
194     class(*), pointer :: res
195     integer :: i
196     type(link),pointer :: alink
197     if (associated(this%firstLink)) then
198         if (this%numLinks>=n) then
199             call this%reset()
200             do i=1, n-1     !iterating trough the list until the desired position
201                 call this%next()
202             end do
203             if (this%moreValues()) then
204             res => this%currLink%get()
205             else
206                 stop '[LinkedList::getValue]: link non-existant, something went wrong!'
207             end if
208             call this%reset()
209         else
210             stop '[LinkedList::getValue]: index out of bounds'
211         end if
212     else
213         stop '[LinkedList::getValue]: list is empty'
214     end if
```

6.1.2.7 morevalues()

pure logical function abstract_linkedlist_mod::morevalues (
class(linkedlist), intent(in) *this*) [private]

Method that returns a logical with signaling if the current link is ok.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 254 of file abstract_LinkedList.f90.

```
254     class(linkedlist), intent(in) :: this
255     morevalues = associated(this%currLink)
```

6.1.2.8 next()

subroutine abstract_linkedlist_mod::next (
class(linkedlist) *this*) [private]

Method that returns the next link in the list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 234 of file abstract_LinkedList.f90.

```
234     class(linkedlist) :: this
235     this%currLink => this%currLink%nextLink()
```

6.1.2.9 previous()

```
subroutine abstract_linkedlist_mod::previous (
            class(linkedlist) this )  [private]
```

Method that returns the previous link in the list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 244 of file abstract_LinkedList.f90.

```
244     class(linkedlist) :: this
245     this%currLink => this%currLink%previousLink()
```

6.1.2.10 remove()

```
subroutine abstract_linkedlist_mod::remove (
            class(linkedlist), intent(inout) this,
            integer, intent(in) n )  [private]
```

Method that removes the nth link from a list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 134 of file abstract_LinkedList.f90.

```
134     class(linkedlist), intent(inout) :: this
135     integer, intent(in) :: n
136     class(link), pointer :: previouslink
137     class(link), pointer :: nextlink
138     integer :: i
139     if (associated(this%firstLink)) then
140         if (this%numLinks>=n) then
141             call this%reset()
142             do i=1, n-1     !iterating trough the list until the desired position
143                 call this%next()
144             end do
145             if (this%moreValues()) then
146                 call this%removeCurrent()
147             end if
148         else
149             stop '[LinkedList::remove]: index out of bounds'
150         end if
151     end if
```

6.1.2.11 removecurrent()

```
subroutine abstract_linkedlist_mod::removecurrent (
            class(linkedlist), intent(inout) this )  [private]
```

Method that removes a link from the list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 104 of file abstract_LinkedList.f90.

```
104     class(linkedlist), intent(inout) :: this
105     class(link), pointer :: previouslink
106     class(link), pointer :: nextlink
107
108     previouslink => this%currLink%previousLink()
109     nextlink => this%currLink%nextLink()
110
111     if (associated(this%currLink,this%firstLink)) then !This is the first link
112         this%firstLink => nextlink
113     end if
114     if (associated(previouslink)) then
115         call previouslink%setNextLink(nextlink)
116     end if
117     if (associated(nextlink)) then
118         call nextlink%setPreviousLink(previouslink)
119     end if
120
121     call this%currLink%removeLink()
122     deallocate(this%currLink)
123     this%currLink => nextlink
124     this%numLinks = this%numLinks - 1
125
```

6.1.2.12 reset()

```
subroutine abstract_linkedlist_mod::reset (
            class(linkedlist) this )  [private]
```

Method that resets the list iterator.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 264 of file abstract_LinkedList.f90.

```
264     class(linkedlist) :: this
265     this%currLink => this%firstLink
```

6.2 aot_mod Module Reference

Module to hold the Arrays of Tracers class and its methods. This class defines a collection of id, xyz, uvw, .. arrays that allow for easy and efficient manipulation of the Tracer objects. These must be exported into the objects from this class.

Data Types

- interface aot
- type aot_class

  Arrays of Tracers class.
- type trc_ptr_class

Functions/Subroutines

- type(aot_class) function constructor (trclist)

  Constructor for AoT object with data from a tracerList_class object.
- subroutine clean (self)

  Destructor for AoT object, deallocates all contents.
- subroutine totracers (self)

  Sends the data on the AoT to the Tracer objects. Less type guard checks because they were already made in the constructor of the AoT.
- subroutine print_aot (self)

  Method that prints all the elements of the array.

6.2.1 Detailed Description

Module to hold the Arrays of Tracers class and its methods. This class defines a collection of id, xyz, uvw, .. arrays that allow for easy and efficient manipulation of the Tracer objects. These must be exported into the objects from this class.

Author

Ricardo Birjukovs Canelas

6.2.2 Function/Subroutine Documentation

6.2.2.1 clean()

```
subroutine aot_mod::clean (
            class(aot_class), intent(inout) self )  [private]
```

Destructor for AoT object, deallocates all contents.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 111 of file AoT.f90.

```
111     implicit none
112     class(aot_class), intent(inout) :: self
113     if (allocated(self%id)) deallocate(self%id)
114     !if (associated(self%trc%ptr)) nullify(self%trc%ptr) !need make sure there are no memory leaks
115     if (allocated(self%trc)) deallocate(self%trc)
116     if (allocated(self%x)) deallocate(self%x)
117     if (allocated(self%y)) deallocate(self%y)
118     if (allocated(self%z)) deallocate(self%z)
119     if (allocated(self%u)) deallocate(self%u)
120     if (allocated(self%v)) deallocate(self%v)
121     if (allocated(self%w)) deallocate(self%w)
```

6.2.2.2 constructor()

```
type(aot_class) function aot_mod::constructor (
            class(tracerlist_class), intent(in) trclist )  [private]
```

Constructor for AoT object with data from a tracerList_class object.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*trclist*	

Definition at line 62 of file AoT.f90.

```
62      implicit none
63      type(aot_class) :: constructor
64      class(tracerList_class), intent(in) :: trclist
65      integer :: nt, i
66      class(*), pointer :: aTracer
67      type(string) :: outext
68      !allocating the necessary space
69      nt = trclist%getSize()
70      allocate(constructor%id(nt))
71      allocate(constructor%trc(nt))
72      allocate(constructor%x(nt))
73      allocate(constructor%y(nt))
74      allocate(constructor%z(nt))
75      allocate(constructor%u(nt))
76      allocate(constructor%v(nt))
77      allocate(constructor%w(nt))
78      nt=1
79      call trclist%reset()                 ! reset list iterator
80      do while(trclist%moreValues())       ! loop while there are values
81          atracer => trclist%currentValue() ! get current value
82          select type(atracer)
83          class is (tracer_class)
84              if (atracer%now%active) then
85                  constructor%id(nt) = atracer%par%id
86                  constructor%trc(nt)%ptr => atracer
87                  constructor%x(nt) = atracer%now%pos%x
88                  constructor%y(nt) = atracer%now%pos%y
89                  constructor%z(nt) = atracer%now%pos%z
90                  constructor%u(nt) = atracer%now%vel%x
91                  constructor%v(nt) = atracer%now%vel%y
92                  constructor%w(nt) = atracer%now%vel%z
93                  nt= nt + 1
94              end if
95              class default
96              outext = '[AoT::Constructor]: Unexepected type of content, not a Tracer'
97              call log%put(outext)
98              stop
99          end select
100          call trclist%next()             ! increment the list iterator
101      end do
102      call trclist%reset()                ! reset list iterator
```

6.2.2.3 print_aot()

```
subroutine aot_mod::print_aot (
            class(aot_class), intent(in) self )  [private]
```

Method that prints all the elements of the array.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 162 of file AoT.f90.

```
162     class(aot_class), intent(in) :: self
163     type(string) :: outext, t(4)
164     integer :: i
165     do i=1, size(self%id)
166         t(1) = self%id(i)
167         t(2) = self%x(i)
168         t(3) = self%y(i)
169         t(4) = self%z(i)
170         outext = 'Tracer['//t(1)//']::xyz('//t(2)//','//t(3)//','//t(4)//')'
171         call log%put(outext,.false.)
172     end do
```

6.2.2.4 totracers()

```
subroutine aot_mod::totracers (
            class(aot_class), intent(in) self )  [private]
```

Sends the data on the AoT to the Tracer objects. Less type guard checks because they were already made in the constructor of the AoT.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 131 of file AoT.f90.

```
131     implicit none
132     class(aot_class), intent(in) :: self
133     integer :: i
134     class(tracer_class), pointer :: aTracer
135     type(string) :: outext
136     if (allocated(self%id)) then
137         do i=1, size(self%id)
138             if (associated(self%trc(i)%ptr)) then
139                 atracer => self%trc(i)%ptr
140                 atracer%now%pos%x = self%x(i)
141                 atracer%now%pos%y = self%y(i)
142                 atracer%now%pos%z = self%z(i)
143                 atracer%now%vel%x = self%u(i)
144                 atracer%now%vel%y = self%v(i)
145                 atracer%now%vel%z = self%w(i)
146             else
147                 outext = '[AoT::AoTtoTracers]: pointer to Tracer no associated, stoping'
148                 call log%put(outext)
149                 stop
150             end if
151         end do
152     end if
```

6.3 background_mod Module Reference

Defines a background class that describes a solution from which to interpolate. A background object contains an arbitrary number of scalar or vectorial fields, in 2, 3 or 4D, indexed to labeled 1D fields of dimensions. The fields are stored in a linked list, enabling trivial iteration.

Data Types

- interface background
- type background_class
- type fieldslist_class

Functions/Subroutines

- subroutine addfield (self, gfield)

    Method that adds a field to the Background object's field list.
- type(background_class) function constructor (id, name, extents, dims)

    Constructor for Background object.
- subroutine setdims (self, dims)

    Method that allocates and sets the dimensions of the Background object.
- subroutine setextents (self, bbox)

    Method that sets the extents (bounding box) of the Background object.
- subroutine setid (self, id, name)

    Method that sets the ID and name of the Background object.
- subroutine test (self)

    A class 'unit' test for the background_class.
- subroutine printbackground (self)

    Method that prints the Background object.
- subroutine print_fieldlist (this)

    Method that prints all the links of the list.
- subroutine print_fieldlistcurrent (this)

    Method that prints the current link of the list.

6.3.1 Detailed Description

Defines a background class that describes a solution from which to interpolate. A background object contains an arbitrary number of scalar or vectorial fields, in 2, 3 or 4D, indexed to labeled 1D fields of dimensions. The fields are stored in a linked list, enabling trivial iteration.

Author

Ricardo Birjukovs Canelas

6.3.2 Function/Subroutine Documentation

6.3.2.1 addfield()

```
subroutine background_mod::addfield (
            class(background_class), intent(inout) self,
            type(generic_field_class), intent(in) gfield )  [private]
```

Method that adds a field to the Background object's field list.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,gfield*	

Definition at line 68 of file background.f90.

```
implicit none
class(background_class), intent(inout) :: self
type(generic_field_class), intent(in) :: gfield
if (allocated(gfield%scalar1d%field)) call self%fields%add(gfield%scalar1d)
if (allocated(gfield%scalar2d%field)) call self%fields%add(gfield%scalar2d)
if (allocated(gfield%scalar3d%field)) call self%fields%add(gfield%scalar3d)
if (allocated(gfield%scalar4d%field)) call self%fields%add(gfield%scalar4d)
if (allocated(gfield%vectorial2d%field)) call self%fields%add(gfield%vectorial2d)
if (allocated(gfield%vectorial3d%field)) call self%fields%add(gfield%vectorial3d)
if (allocated(gfield%vectorial4d%field)) call self%fields%add(gfield%vectorial4d)
```

6.3.2.2 constructor()

```
type(background_class) function background_mod::constructor (
            integer, intent(in) id,
            type(string), intent(in) name,
            type(box), intent(in) extents,
            type(scalar1d_field_class), dimension(:), intent(in) dims )  [private]
```

Constructor for Background object.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*id,name,extents,dims*	

Definition at line 87 of file background.f90.

```
implicit none
type(background_class) :: constructor
integer, intent(in) :: id
type(string), intent(in) :: name
type(box), intent(in) :: extents
type(scalar1d_field_class), dimension(:), intent(in) :: dims
call constructor%setID(id, name)
call constructor%setExtents(extents)
call constructor%setDims(dims)
```

6.3.2.3 print_fieldlist()

```
subroutine background_mod::print_fieldlist (
            class(fieldslist_class), intent(in) this )  [private]
```

Method that prints all the links of the list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 210 of file background.f90.

```
210     class(fieldsList_class), intent(in) :: this
211     class(*), pointer :: curr
212     call this%reset()                ! reset list iterator
213     do while(this%moreValues())      ! loop while there are values to print
214         call this%printCurrent()
215         call this%next()             ! increment the list iterator
216     end do
217     call this%reset()                ! reset list iterator
```

6.3.2.4 print_fieldlistcurrent()

subroutine background_mod::print_fieldlistcurrent (
class(fieldslist_class), intent(in) *this*) [private]

Method that prints the current link of the list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 226 of file background.f90.

```
226     class(fieldsList_class), intent(in) :: this
227     class(*), pointer :: curr
228     type(string) :: outext
229     curr => this%currentValue() ! get current value
230     select type(curr)
231     class is (field_class)
232         call curr%print()
233         class default
234         outext = '[fieldsList_class::print] Unexepected type of content, not a Field'
235         call log%put(outext)
236         stop
237     end select
```

6.3.2.5 printbackground()

subroutine background_mod::printbackground (
class(background_class), intent(inout) *self*) [private]

Method that prints the Background object.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 187 of file background.f90.

```
187     class(background_class), intent(inout) :: self
188     type(string) :: outext, t
189     integer :: i
190     t = self%id
191     outext = 'Background['//t//', '//self%name//'] is a'
192     call log%put(outext,.false.)
193     call geometry%print(self%extents)
194     outext = 'The dimensions fields are:'
195     call log%put(outext,.false.)
196     do i=1, size(self%dim)
197         call self%dim(i)%print()
198     end do
199     outext = 'The data fields are:'
200     call log%put(outext,.false.)
201     call self%fields%print()
```

6.3.2.6 setdims()

```
subroutine background_mod::setdims (
            class(background_class), intent(inout) self,
            type(scalar1d_field_class), dimension(:), intent(in) dims )  [private]
```

Method that allocates and sets the dimensions of the Background object.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,dims*	

Definition at line 105 of file background.f90.

```
105     class(background_class), intent(inout) :: self
106     type(scalar1d_field_class), dimension(:), intent(in) :: dims
107     allocate(self%dim, source = dims)
```

6.3.2.7 setextents()

```
subroutine background_mod::setextents (
            class(background_class), intent(inout) self,
            type(box), intent(in) bbox )  [private]
```

Method that sets the extents (bounding box) of the Background object.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,bbox*	

Definition at line 117 of file background.f90.

```
117     class(background_class), intent(inout) :: self
118     type(box), intent(in) :: bbox
119     self%extents = bbox
```

6.3.2.8 setid()

```
subroutine background_mod::setid (
            class(background_class), intent(inout) self,
            integer, intent(in) id,
            type(string), intent(in) name )  [private]
```

Method that sets the ID and name of the Background object.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,id,name*	

Definition at line 129 of file background.f90.

```
129     class(background_class), intent(inout) :: self
130     integer, intent(in) :: id
131     type(string), intent(in) :: name
132     self%id = id
133     self%name = name
```

6.3.2.9 test()

```
subroutine background_mod::test (
            class(background_class), intent(inout) self )  [private]
```

A class 'unit' test for the background_class.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 142 of file background.f90.

```
142     class(background_class), intent(inout) :: self
143     type(background_class) :: background1
144     type(generic_field_class) :: gfield1, gfield2, gfield3
145     real(prec), allocatable, dimension(:) :: field1
146     real(prec), allocatable, dimension(:,:) :: field2
147     type(vector), allocatable, dimension(:,:,:) :: field3
148     type(string) :: name1, name2, name3, bname
149     type(string) :: units1, units2, units3
150     type(box) :: backgroundbbox
151     type(scalar1d_field_class), allocatable, dimension(:) :: backgroundims
152     !generating fields
153     allocate(field1(50))
154     allocate(field2(20,60))
155     allocate(field3(2,3,4))
156     name1 = 'testfield1d'
157     name2 = 'testfield2d'
158     name3 = 'testfield3d'
159     units1 = 'm/s'
160     units2 = 'km'
```

```
units3 = 'ms-1'
call gfield1%initialize(name1, units1, field1)
call gfield2%initialize(name2, units2, field2)
call gfield3%initialize(name3, units3, field3)
!assembling our Background
bname = 'TestBackground'
name1 = 'lon'
name2 = 'lat'
backgroundbbox%pt = 1*ex + 2*ey + 3*ez
backgroundbbox%size = 4*ex + 5*ey + 6*ez
allocate(backgroundims(2))
call backgroundims(1)%initialize(name1,units2,1, field1)
call backgroundims(2)%initialize(name2,units2,1, field1)
background1 = background(5, bname, backgroundbbox, backgroundims)
call background1%add(gfield1)
call background1%add(gfield2)
call background1%add(gfield3)
call background1%print()
```

6.4 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

- integer function numallocTracers (self)

  method that returns the total allocated Tracers in the Block
- subroutine initblock (self, id, templatebox)

  method to allocate and initialize Blocks and their Emitters
- subroutine putsource (self, sourcetoadd)

  Method to place a Source on the Block sourceList_class object. Adds the Source info to the Block Emitter.
- subroutine toogleblocksources (self)

  Method to activate and deactivate the sources on this block, based on GlobaSimTime.
- subroutine callemitter (self)

  Method to emitt Tracers from currently active Sources on the Block.
- subroutine distributetracers (self)

  Method to distribute the Tracers to their correct Blocks.
- subroutine consolidatearrays (self)

  Method to clean the Tracer list from inactive Tracers. TODO test further optimization.
- subroutine tracerstoaot (self)

  Method to build the AoT object at this timestep for actual numerical work.
- subroutine aottotracers (self)

  Method to write the data in the AoT back to the Tracer objects in the list.
- subroutine cleanaot (self)

  Method to clean out the AoT object.
- subroutine sendtracer (blk, trc)

  Method to send a Tracer from the current Block to another Block.
- integer function, public getblockindex (pt)

  Returns the index of a Block for a given set of coordinates.

- 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

6.4.1 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

6.4.2 Function/Subroutine Documentation

6.4.2.1 allocblocks()

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

routine to allocate the simulation blocks

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

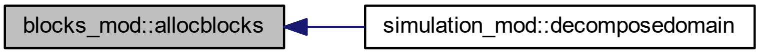
in	*nblk*	

Definition at line 436 of file blocks.f90.

```
436     implicit none
437     integer, intent(in) ::  nblk
438     type(string) :: outext, temp
```

```
    integer err
    allocate(dblock(nblk), stat=err)
    if(err/=0)then
        outext='[allocBlobks]: Cannot allocate Blocks, stoping'
        call log%put(outext)
        stop
    else
        temp = nblk
        outext = 'Allocated '// temp // ' Blocks.'
        call log%put(outext)
    endif
```

Here is the caller graph for this function:



6.4.2.2 aottotracers()

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

Method to write the data in the AoT back to the Tracer objects in the list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 287 of file blocks.f90.

```
287     implicit none
288     class(block_class), intent(inout) :: self
289     call self%AoT%toTracers()
```

6.4.2.3 callemitter()

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

Method to emitt Tracers from currently active Sources on the Block.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 164 of file blocks.f90.

```
164     implicit none
165     class(block_class), intent(inout) :: self
166     integer :: i
167     class(*), pointer :: aSource
168     type(string) :: outext
169 
170     call self%LSource%reset()                   ! reset list iterator
171     do while(self%LSource%moreValues())         ! loop while there are values
172         asource => self%LSource%currentValue()  ! get current value
173         select type(asource)
174         class is (source_class)
175             if (asource%now%active) then
176                 asource%now%emission_stride = asource%now%emission_stride - 1   !decreasing the stride at this dt
177                 if (asource%now%emission_stride == 0) then                      !reached the bottom of the stride stack, time to emitt
178                     call self%Emitter%emitt(asource, self%LTracer)
179                     asource%now%emission_stride = asource%par%emitting_rate     !reseting the stride after the Source emitts
180                 end if
181             end if
182             class default
183             outext = '[Block::CallEmitter] Unexepected type of content, not a Source'
184             call log%put(outext)
185             stop
186         end select
187         call self%LSource%next()            ! increment the list iterator
188     end do
189     call self%LSource%reset()               ! reset list iterator
190 
```

6.4.2.4 cleanaot()

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

Method to clean out the AoT object.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 298 of file blocks.f90.

```
298     implicit none
299     class(block_class), intent(inout) :: self
300     call self%AoT%Clean()
```

6.4.2.5 consolidatearrays()

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

Method to clean the Tracer list from inactive Tracers. TODO test further optimization.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 239 of file blocks.f90.

```
239     implicit none
240     class(block_class), intent(inout) :: self
241     class(*), pointer :: aTracer
242     type(string) :: outext
243     logical :: notremoved
244 
245     call self%LTracer%reset()                    ! reset list iterator
246     do while(self%LTracer%moreValues())          ! loop while there are values
247         notremoved = .true.
248         atracer => self%LTracer%currentValue()   ! get current value
249         select type(atracer)
250         class is (tracer_class)
251             if (atracer%now%active.eqv. .false.) then
252                 call self%LTracer%removeCurrent() !this advances the iterator to the next position
253                 notremoved = .false.
254             end if
255             class default
256             outext = '[Block::ConsolidateArrays]: Unexepected type of content, not a Tracer'
257             call log%put(outext)
258             stop
259         end select
260         if (notremoved) call self%LTracer%next()    ! increment the list iterator
261     end do
262     call self%LTracer%reset()                       ! reset list iterator
263 
```

6.4.2.6 distributetracers()

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

Method to distribute the Tracers to their correct Blocks.

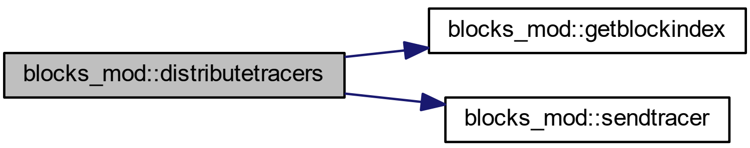
Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 199 of file blocks.f90.

```
    implicit none
    class(block_class), intent(inout) :: self
    integer :: i, blk
    class(*), pointer :: aTracer
    type(string) :: outext
    logical :: notremoved

    call self%LTracer%reset()                    ! reset list iterator
    do while(self%LTracer%moreValues())          ! loop while there are values
        notremoved = .true.
        atracer => self%LTracer%currentValue()   ! get current value
        select type(atracer)
        class is (tracer_class)
            if (atracer%now%active) then
                blk = getblockindex(atracer%now%pos)
                if (blk /= self%id) then         !tracer is on a different block than the current one
                    !PARALLEL this is a CRITICAL section, need to ensure correct tracer index attribution
                    call sendtracer(blk,atracer)
                    call self%LTracer%removeCurrent() !this also advances the iterator to the next position
                    notremoved = .false.
                end if
            end if
        class default
            outext = '[Block::DistributeTracers]: Unexepected type of content, not a Tracer'
            call log%put(outext)
            stop
        end select
        if (notremoved) call self%LTracer%next()     ! increment the list iterator
    end do
    call self%LTracer%reset()                    ! reset list iterator

```

Here is the call graph for this function:



6.4.2.7 getblockindex()

```
integer function, public blocks_mod::getblockindex (
            type(vector), intent(in) pt )
```

Returns the index of a Block for a given set of coordinates.

Author

Ricardo Birjukovs Canelas - MARETEC

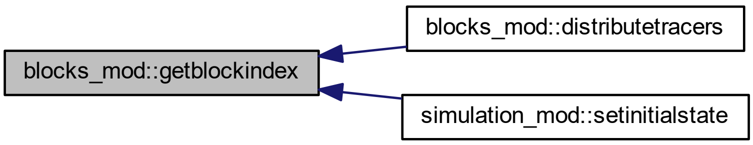
Parameters

in	*pt*	

Definition at line 324 of file blocks.f90.

```
324     implicit none
325     type(vector), intent(in) :: pt
326     integer :: ix, iy, temp
327     type(string) :: outext
328     ix = min(int((pt%x + bbox%offset%x)/globals%SimDefs%blocksize%x) + 1, globals%SimDefs%numblocksx)
329     iy = min(int((pt%y + bbox%offset%y)/globals%SimDefs%blocksize%y) + 1, globals%SimDefs%numblocksy)
330     temp = 2*ix + iy -2
331     if (temp > globals%SimDefs%numblocks) then
332         outext='[Blocks::getBlockIndex]: problem in getting correct Block index, stoping'
333         call log%put(outext)
334         stop
335     end if
336     getblockindex = temp
```

Here is the caller graph for this function:



6.4.2.8 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,id,templatebox*	

Definition at line 91 of file blocks.f90.

```
91     implicit none
92     class(block_class), intent(inout) :: self
93     integer, intent(in) :: id
94     type(box), intent(in) :: templatebox
95     integer :: sizem
96     self%id = id
```

```
!setting the block sub-domain
self%extents%pt = templatebox%pt
self%extents%size = templatebox%size
 !initializing the block emitter
 call self%Emitter%initialize()
 sizem = sizeof(self)
 call simmemory%addblock(sizem)
```

6.4.2.9 numalloctracers()

```
integer function blocks_mod::numalloctracers (
            class(block_class), intent(in) self )  [private]
```

method that returns the total allocated Tracers in the Block

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 78 of file blocks.f90.

```
78      implicit none
79      class(block_class), intent(in) :: self
80      integer :: numAllocTracers
81      numalloctracers = self%LTracer%getSize()
```

6.4.2.10 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 346 of file blocks.f90.

```
346      implicit none
347      class(block_class), intent(inout) :: self
348      type(string) :: outext, temp_str
349      temp_str = self%id
350      outext='-->Block '//temp_str//' is a'
351      call log%put(outext,.false.)
```

```
352     call geometry%print(self%extents)
353     temp_str = self%LSource%getSize()
354     outext='      and has '//temp_str//' Sources'
355     call log%put(outext,.false.)
```

6.4.2.11 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 365 of file blocks.f90.

```
365     implicit none
366     class(block_class), intent(inout) :: self
367     type(string) :: outext, temp_str
368     integer :: i
369     temp_str = self%id
370     outext='-->Block '//temp_str//' is a'
371     call log%put(outext,.false.)
372     call geometry%print(self%extents)
373     temp_str = self%LSource%getSize()
374     outext='      and has '//temp_str//' Sources'
375     call log%put(outext,.false.)
376     call self%LSource%print()
```

6.4.2.12 putsource()

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

Method to place a Source on the Block sourceList_class object. Adds the Source info to the Block Emitter.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,sourcetoadd*	
in,out	*sourcetoadd*	Source object to store

Definition at line 114 of file blocks.f90.

```
114     implicit none
115     class(block_class), intent(inout) :: self
116     class(source_class), intent(inout) :: sourcetoadd
117     call self%LSource%add(sourcetoadd)
118     !adding this Source to the Block Emitter pool
119     call self%Emitter%addSource(sourcetoadd)
```

6.4.2.13 sendtracer()

```
subroutine blocks_mod::sendtracer (
            integer, intent(in) blk,
            class(*), intent(in) trc )  [private]
```

Method to send a Tracer from the current Block to another Block.

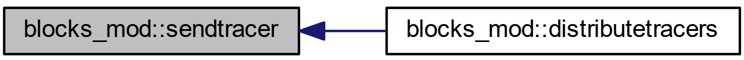
Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 309 of file blocks.f90.

```
309     implicit none
310     integer, intent(in) :: blk
311     class(*), intent(in) :: trc
312     !PARALLEL this is a CRITICAL section, need to ensure correct tracer
313     !index attribution at the new block
314     call dblock(blk)%LTracer%add(trc)
```

Here is the caller graph for this function:



6.4.2.14 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

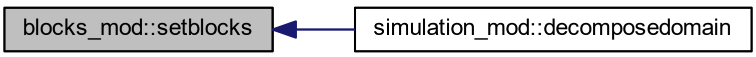
Parameters

in	*auto,nblk,nxi,nyi*	

Definition at line 386 of file blocks.f90.

```
386     implicit none
387     logical, intent(in) ::  auto
388     integer, intent(in) ::  nblk
389     integer, intent(out) :: nxi, nyi
390     type(string) :: outext, temp(2)
391     integer :: i, j, b
392     real(prec) :: ar
393     type(box) :: tempbox
394
395     if (auto) then
396         ar = bbox%size%x/bbox%size%y
397         ar = get_closest_twopow(ar) !aspect ratio of our bounding box
398         nyi = sqrt(nblk/ar)
399         if (nyi == 0) then
400             temp(1) = ar
401             outext='[setBlocks]: block auto sizing failed. Bouding box aspect ratio = '//temp(1)//'. Stoping'
402             call log%put(outext)
403             stop
404         endif
405         nxi = (nblk/nyi)
406
407         b=1
408         do i=1, nxi
409             do j=1, nyi
410                 tempbox%pt = bbox%pt + bbox%size%x*(i-1)/nxi*ex + bbox%size%y*(j-1)/nyi*ey - bbox%pt%z*ez
411                 tempbox%size = bbox%size%x/nxi*ex + bbox%size%y/nyi*ey
412                 call dblock(b)%initialize(b, tempbox)
413                 b=b+1
414             end do
415         end do
416         temp(1) = nxi
417         temp(2) = nyi
418         outext='-->Automatic domain decomposition sucessful. Domain is '//temp(1)// ' X ' //temp(2)//' Blocks'
419         call log%put(outext,.false.)
420     end if
421     globals%SimDefs%blocksize = dblock(1)%extents%size
422     !do i=1, size(DBlock)
423     !    call DBlock(i)%print()
424     !enddo
425
426     return
```

Here is the caller graph for this function:



6.4.2.15 toogleblocksources()

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

Method to activate and deactivate the sources on this block, based on GlobaSimTime.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 129 of file blocks.f90.

```
    implicit none
    class(block_class), intent(inout) :: self
    integer :: i
    class(*), pointer :: aSource
    type(string) :: outext

    call self%LSource%reset()                   ! reset list iterator
    do while(self%LSource%moreValues())         ! loop while there are values
        asource => self%LSource%currentValue()  ! get current value
        select type(asource)
        class is (source_class)
            if (globals%SimTime <= asource%par%stoptime) then       !SimTime smaller than Source end time
                if (globals%SimTime >= asource%par%starttime) then   !SimTime larger than source start time
                    asource%now%active = .true.
                end if
            else            !SimTime larger than Source end time
                asource%now%active = .false.
            end if
            class default
            outext = '[Block::ToogleBlockSources] Unexepected type of content, not a Source'
            call log%put(outext)
            stop
        end select
        call self%LSource%next()            ! increment the list iterator
    end do
    call self%LSource%reset()               ! reset list iterator
```

6.4.2.16 tracerstoaot()

subroutine blocks_mod::tracerstoaot (
class(block_class), intent(inout) *self*) [private]

Method to build the AoT object at this timestep for actual numerical work.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 272 of file blocks.f90.

```
    implicit none
    class(block_class), intent(inout) :: self
    self%AoT = aot(self%LTracer)
    !if (self%LTracer%getSize() > 0) then
    !    print*, 'From Block ', self%id
    !    call self%AoT%print()
    !end if
```

6.4.3 Variable Documentation

6.4.3.1 dblock

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

Definition at line 63 of file blocks.f90.

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

6.5 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

6.5.1 Detailed Description

Module that defines a simulation Bounding Box.

Author

Ricardo Birjukovs Canelas

6.5.2 Function/Subroutine Documentation

6.5.2.1 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.

```
45      implicit none
46      class(boundingbox_class), intent(inout) :: self
47      self%pt = globals%SimDefs%Pointmin
48      !self%size = geo2m(Globals%SimDefs%Pointmax - Globals%SimDefs%Pointmin, Globals%SimDefs%Pointmin%y)
49      self%size = globals%SimDefs%Pointmax - globals%SimDefs%Pointmin
50      self%offset = -self%pt !distance to the origin - local reference
```

6.5.2.2 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 59 of file boundingbox.f90.

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

6.5.3 Variable Documentation

6.5.3.1 bbox

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

Definition at line 33 of file boundingbox.f90.

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

6.6 common_modules Module Reference

Module to hold all of the commonly used base modules.

6.6.1 Detailed Description

Module to hold all of the commonly used base modules.

Author

Ricardo Birjukovs Canelas

6.7 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 deletecontent (this)

  Method that deletes the value 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.

6.7.1 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

6.7.2 Function/Subroutine Documentation

6.7.2.1 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

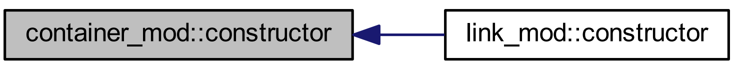
Parameters

in	*to_store*	

Definition at line 120 of file container.f90.

```
120     class(container), pointer :: constructor
121     class(*), intent(in) :: to_store
122     allocate(constructor)
123     allocate(constructor%value, source=to_store)
```

Here is the caller graph for this function:



6.7.2.2 deletecontent()

```
subroutine container_mod::deletecontent (
            class(container), intent(inout) this )  [private]
```

Method that deletes the value in the container.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 74 of file container.f90.

```
74     class(container), intent(inout) :: this
75     deallocate(this%value)
```

6.7.2.3 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

in	*this*	

Definition at line 63 of file container.f90.

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

6.7.2.4 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

in	*this*	

Definition at line 99 of file container.f90.

```
99      class(container), intent(in) :: this
100      select type(v => this%value)
101      type is (integer)
102          print *, v
103      type is (character(*))
104          print *, v(1:1)
105      type is (real)
106          print *, v
107          class default
108          print*, "[printContainer]: don't know how to print this value, ignoring"
109      end select
```

6.7.2.5 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

in	*this,to_store*	

Definition at line 86 of file container.f90.

```
86      class(container), intent(inout) :: this
87      class(*), intent(in) :: to_store
88      allocate(this%value, source=to_store)
```

6.8 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 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 that space in the Blocks Tracer array
- subroutine removesource (self, src)

    method to remove from the total emittable particles count a Source
- subroutine class(source_class), intent(inout) emitt (self, src, trclist)

    method that emitts the Tracers, based on the Sources on this Block Emitter
- subroutine tracermaker (self, trc, src, p)

    method that calls the corresponding Tracer constructor, depending on the requested type from the emitting Source

6.8.1 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

6.8.2 Function/Subroutine Documentation

6.8.2.1 addsource()

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

method to compute the total emittable particles per source and allocate that space in the Blocks Tracer array

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,src*	

Definition at line 68 of file emitter.f90.

```
68     implicit none
69     class(emitter_class), intent(inout) :: self
70     class(source_class),intent(in) :: src
71     self%emittable = self%emittable + src%stencil%total_np
```

6.8.2.2 emitt()

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

method that emitts the Tracers, based on the Sources on this Block Emitter

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,src,trclist*	

Definition at line 94 of file emitter.f90.

```
94     implicit none
95     class(emitter_class), intent(inout) :: self
96     class(source_class), intent(inout)  :: src
97     class(tracerList_class), intent(inout)   :: trclist
98     integer i
99     class(*), allocatable :: newtrc
100     do i=1, src%stencil%np
101         self%emitted = self%emitted + 1
102         !PARALLEL The calls inside this routine MUST be atomic in order to get the correct sequencial Tracer Id
103         call self%tracerMaker(newtrc, src, i)
104         call trclist%add(newtrc)
105     end do
106     src%stats%particles_emitted = src%stats%particles_emitted + src%stencil%np
```

6.8.2.3 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

Definition at line 54 of file emitter.f90.

```
54     implicit none
55     class(emitter_class), intent(inout) :: self
56     self%emitted = 0
57     self%emittable = 0
```

6.8.2.4 removesource()

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

method to remove from the total emittable particles count a Source

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,src*	

Definition at line 81 of file emitter.f90.

```
81     implicit none
82     class(emitter_class), intent(inout) :: self
83     class(source_class),intent(in) :: src
84     self%emittable = self%emittable - src%stencil%total_np
```

6.8.2.5 tracermaker()

```
subroutine emitter_mod::tracermaker (
            class(emitter_class), intent(in) self,
            class(*), intent(out), allocatable trc,
            class(source_class), intent(in) src,
            integer, intent(in) p ) [private]
```

method that calls the corresponding Tracer constructor, depending on the requested type from the emitting Source

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,trc,src,p*	

Definition at line 117 of file emitter.f90.

```
117    implicit none
118    class(emitter_class), intent(in) :: self
119    class(*), allocatable, intent(out) :: trc
120    class(source_class), intent(in) :: src
121    integer, intent(in) :: p
122    type(string) :: outext, temp
123
```

```
    !PARALLEL Globals%Sim%getnumTracer() MUST be atomic in order to get the correct sequencial Tracer Id
    select case (src%prop%property_type%chars())
    case ('base')
        allocate(trc, source = tracer(globals%Sim%getnumTracer(), src, globals%SimTime, p)) !Beacause ifort
   is not F2008 compliant.
        !trc = Tracer(1, src, Globals%SimTime, p) !Otherwise instinsic allocation would be enough and more
   readable, like this. Compiles fine in GFortran
    case ('paper')
        allocate(trc, source = papertracer(globals%Sim%getnumTracer(), src, globals%SimTime, p))
    case ('plastic')
        allocate(trc, source = tracer(globals%Sim%getnumTracer(), src, globals%SimTime, p))
        case default
        outext='[Emitter::tracerMaker]: unexpected type for Tracer object: '//src%prop%property_type
        call log%put(outext)
        stop
    end select

```

6.9 field_types_mod Module Reference

Defines classes for 'fields': 1, 2, 3 and 4D labeled data. Valid for both scalar and vectorial (real) data. Defines a generic wrapper for these classes, that abstracts the user from having to choose their data dimensionality or type to create a field.

Data Types

- type field_class
- type generic_field_class

  generic field class. This works as a wrapper for a generic initialization routine.
- type scalar1d_field_class

  a 1D scalar field class
- type scalar2d_field_class

  a 2D scalar field class
- type scalar3d_field_class

  a 3D scalar field class
- type scalar4d_field_class

  a 4D scalar field class
- type scalar_field_class

  a scalar field class
- type vectorial2d_field_class

  a 2D vectorial field class
- type vectorial3d_field_class

  a 3D vectorial field class
- type vectorial4d_field_class

  a 4D vectorial field class
- type vectorial_field_class

  a vectorial field class

Functions/Subroutines

- subroutine inits1d (self, name, units, field)

    Method that allocates and initializes a scalar 1D field in a generic field.
- subroutine inits2d (self, name, units, field)

    Method that allocates and initializes a scalar 2D field in a generic field.
- subroutine inits3d (self, name, units, field)

    Method that allocates and initializes a scalar 3D field in a generic field.
- subroutine inits4d (self, name, units, field)

    Method that allocates and initializes a scalar 4D field in a generic field.
- subroutine initv2d (self, name, units, field)

    Method that allocates and initializes a vectorial 2D field in a generic field.
- subroutine initv3d (self, name, units, field)

    Method that allocates and initializes a vectorial 3D field in a generic field.
- subroutine initv4d (self, name, units, field)

    Method that allocates and initializes a vectorial 4D field in a generic field.
- subroutine initscalar1dfield (self, name, units, dim, field)

    Method that initializes a scalar 1D field.
- subroutine initscalar2dfield (self, name, units, dim, field)

    Method that initializes a scalar 2D field.
- subroutine initscalar3dfield (self, name, units, dim, field)

    Method that initializes a scalar 3D field.
- subroutine initscalar4dfield (self, name, units, dim, field)

    Method that initializes a scalar 4D field.
- subroutine initvectorial2dfield (self, name, units, dim, field)

    Method that initializes a vectorial 2D field.
- subroutine initvectorial3dfield (self, name, units, dim, field)

    Method that initializes a vectorial 3D field.
- subroutine initvectorial4dfield (self, name, units, dim, field)

    Method that initializes a vectorial 4D field.
- subroutine setfieldmetadata (self, name, units, dim)

    Method that initializes a base field object by filling metadata.
- subroutine printgenericfield (self)

    Method that prints the generic field information.
- subroutine test (self)

    A class 'unit' test for the generic_field_class.
- subroutine printfield (self)

    Method that prints the field information.
- type(string) function getfieldtype (self)

    Method that returns the field type (scalar or vectorial), in a string.

6.9.1 Detailed Description

Defines classes for 'fields': 1, 2, 3 and 4D labeled data. Valid for both scalar and vectorial (real) data. Defines a generic wrapper for these classes, that abstracts the user from having to choose their data dimensionality or type to create a field.

Author

Ricardo Birjukovs Canelas

6.9.2 Function/Subroutine Documentation

6.9.2.1 getfieldtype()

```
type(string) function field_types_mod::getfieldtype (
            class(field_class), intent(in) self )  [private]
```

Method that returns the field type (scalar or vectorial), in a string.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 448 of file fields_types.f90.

```
448     class(field_class), intent(in) :: self
449     type(string) :: getFieldType
450     type(string) :: outext
451     select type(self)
452     class is (scalar_field_class)
453         getfieldtype = 'Scalar'
454     class is (vectorial_field_class)
455         getfieldtype = 'Vectorial'
456         class default
457         outext = '[field_class::getFieldType]: Unexepected type of content, not a scalar or vectorial Field'
458         call log%put(outext)
459         stop
460     end select
```

6.9.2.2 inits1d()

```
subroutine field_types_mod::inits1d (
            class(generic_field_class), intent(inout) self,
            type(string), intent(in) name,
            type(string), intent(in) units,
            real(prec), dimension(:), intent(in) field )  [private]
```

Method that allocates and initializes a scalar 1D field in a generic field.

Author

Ricardo Birjukovs Canelas - MARETEC

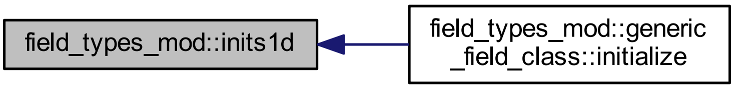
Parameters

in	*self,name,units,field*	

Definition at line 134 of file fields_types.f90.

```
    class(generic_field_class), intent(inout) :: self
    real(prec), intent(in), dimension(:) :: field
    type(string), intent(in) :: name
    type(string), intent(in) :: units
    if (allocated(self%scalar1d%field)) then
        stop '[generic_field_class::initialize]: scalar 1D field already allocated'
    else
        call self%scalar1d%initialize(name, units, 1, field)
    end if
```

Here is the caller graph for this function:



6.9.2.3 inits2d()

```
subroutine field_types_mod::inits2d (
            class(generic_field_class), intent(inout) self,
            type(string), intent(in) name,
            type(string), intent(in) units,
            real(prec), dimension(:,:), intent(in) field )  [private]
```

Method that allocates and initializes a scalar 2D field in a generic field.

Author

Ricardo Birjukovs Canelas - MARETEC

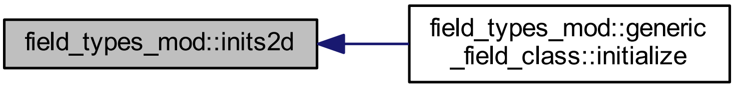
Parameters

in	*self,name,units,field*	

Definition at line 152 of file fields_types.f90.

```
152     class(generic_field_class), intent(inout) :: self
153     real(prec), intent(in), dimension(:,:) :: field
154     type(string), intent(in) :: name
155     type(string), intent(in) :: units
156     if (allocated(self%scalar2d%field)) then
157         stop '[generic_field_class::initialize]: scalar 2D field already allocated'
158     else
159         call self%scalar2d%initialize(name, units, 2, field)
160     end if
```

Here is the caller graph for this function:



6.9.2.4 inits3d()

```
subroutine field_types_mod::inits3d (
            class(generic_field_class), intent(inout) self,
            type(string), intent(in) name,
            type(string), intent(in) units,
            real(prec), dimension(:,:,:), intent(in) field )  [private]
```

Method that allocates and initializes a scalar 3D field in a generic field.

Author

Ricardo Birjukovs Canelas - MARETEC

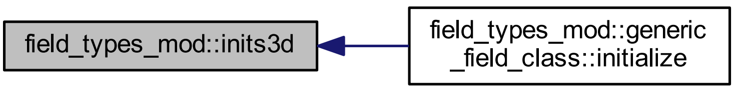
Parameters

in	*self,name,units,field*	

Definition at line 170 of file fields_types.f90.

```
170     class(generic_field_class), intent(inout) :: self
171     real(prec), intent(in), dimension(:,:,:) :: field
172     type(string), intent(in) :: name
173     type(string), intent(in) :: units
174     if (allocated(self%scalar3d%field)) then
175         stop '[generic_field_class::initialize]: scalar 3D field already allocated'
176     else
177         call self%scalar3d%initialize(name, units, 3, field)
178     end if
```

Here is the caller graph for this function:



6.9.2.5 inits4d()

```
subroutine field_types_mod::inits4d (
            class(generic_field_class), intent(inout) self,
            type(string), intent(in) name,
            type(string), intent(in) units,
            real(prec), dimension(:,:,:,:), intent(in) field ) [private]
```

Method that allocates and initializes a scalar 4D field in a generic field.

Author

Ricardo Birjukovs Canelas - MARETEC

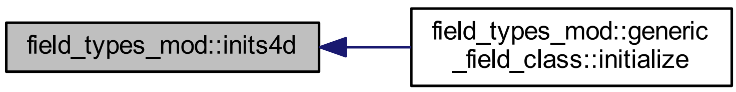
Parameters

in	*self,name,units,field*	

Definition at line 188 of file fields_types.f90.

```
188     class(generic_field_class), intent(inout) :: self
189     real(prec), intent(in), dimension(:,:,:,:) :: field
190     type(string), intent(in) :: name
191     type(string), intent(in) :: units
192     if (allocated(self%scalar4d%field)) then
193         stop '[generic_field_class::initialize]: scalar 4D field already allocated'
194     else
195         call self%scalar4d%initialize(name, units, 4, field)
196     end if
```

Here is the caller graph for this function:



6.9.2.6 initscalar1dfield()

```
subroutine field_types_mod::initscalar1dfield (
            class(scalar1d_field_class), intent(inout) self,
            type(string), intent(in) name,
            type(string), intent(in) units,
            integer, intent(in) dim,
            real(prec), dimension(:), intent(in) field ) [private]
```

Method that initializes a scalar 1D field.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,name,units,dim,field*	

Definition at line 260 of file fields_types.f90.

```
260     class(scalar1d_field_class), intent(inout) :: self
261     real(prec), intent(in), dimension(:) :: field
262     type(string), intent(in) :: name
263     type(string), intent(in) :: units
264     integer, intent(in) :: dim
265     call self%setFieldMetadata(name, units, dim)
266     allocate(self%field, source = field)
```

6.9.2.7 initscalar2dfield()

```
subroutine field_types_mod::initscalar2dfield (
            class(scalar2d_field_class), intent(inout) self,
            type(string), intent(in) name,
            type(string), intent(in) units,
            integer, intent(in) dim,
            real(prec), dimension(:,:), intent(in) field )  [private]
```

Method that initializes a scalar 2D field.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,name,units,dim,field*	

Definition at line 276 of file fields_types.f90.

```
276     class(scalar2d_field_class), intent(inout) :: self
277     real(prec), intent(in), dimension(:,:) :: field
278     type(string), intent(in) :: name
279     type(string), intent(in) :: units
280     integer, intent(in) :: dim
281     call self%setFieldMetadata(name, units, dim)
282     allocate(self%field, source = field)
```

6.9.2.8 initscalar3dfield()

```
subroutine field_types_mod::initscalar3dfield (
            class(scalar3d_field_class), intent(inout) self,
            type(string), intent(in) name,
            type(string), intent(in) units,
            integer, intent(in) dim,
            real(prec), dimension(:,:,:), intent(in) field )  [private]
```

Method that initializes a scalar 3D field.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,name,units,dim,field*	

Definition at line 292 of file fields_types.f90.

```
class(scalar3d_field_class), intent(inout) :: self
real(prec), intent(in), dimension(:,:,:) :: field
type(string), intent(in) :: name
type(string), intent(in) :: units
integer, intent(in) :: dim
call self%setFieldMetadata(name, units, dim)
allocate(self%field, source = field)
```

6.9.2.9 initscalar4dfield()

```
subroutine field_types_mod::initscalar4dfield (
            class(scalar4d_field_class), intent(inout) self,
            type(string), intent(in) name,
            type(string), intent(in) units,
            integer, intent(in) dim,
            real(prec), dimension(:,:,:,:), intent(in) field )  [private]
```

Method that initializes a scalar 4D field.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,name,units,dim,field*	

Definition at line 308 of file fields_types.f90.

```
class(scalar4d_field_class), intent(inout) :: self
real(prec), intent(in), dimension(:,:,:,:) :: field
type(string), intent(in) :: name
type(string), intent(in) :: units
integer, intent(in) :: dim
call self%setFieldMetadata(name, units, dim)
allocate(self%field, source = field)
```

6.9.2.10 initv2d()

```
subroutine field_types_mod::initv2d (
            class(generic_field_class), intent(inout) self,
            type(string), intent(in) name,
            type(string), intent(in) units,
            type(vector), dimension(:,:), intent(in) field )  [private]
```

Method that allocates and initializes a vectorial 2D field in a generic field.

Author

Ricardo Birjukovs Canelas - MARETEC

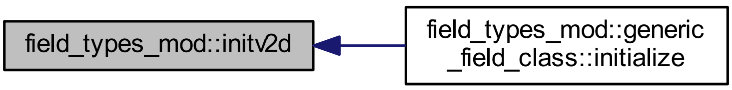
Parameters

in	*self,name,units,field*	

Definition at line 206 of file fields_types.f90.

```
206     class(generic_field_class), intent(inout) :: self
207     type(vector), intent(in), dimension(:,:) :: field
208     type(string), intent(in) :: name
209     type(string), intent(in) :: units
210     if (allocated(self%vectorial2d%field)) then
211         stop '[generic_field_class::initialize]: vectorial 2D field already allocated'
212     else
213         call self%vectorial2d%initialize(name, units, 2, field)
214     end if
```

Here is the caller graph for this function:



6.9.2.11 initv3d()

```
subroutine field_types_mod::initv3d (
            class(generic_field_class), intent(inout) self,
            type(string), intent(in) name,
            type(string), intent(in) units,
            type(vector), dimension(:,:,:), intent(in) field )  [private]
```

Method that allocates and initializes a vectorial 3D field in a generic field.

Author

Ricardo Birjukovs Canelas - MARETEC

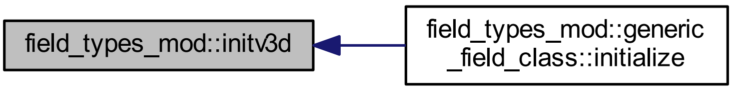
Parameters

in	*self,name,units,field*	

Definition at line 224 of file fields_types.f90.

```
224     class(generic_field_class), intent(inout) :: self
225     type(vector), intent(in), dimension(:,:,:) :: field
226     type(string), intent(in) :: name
227     type(string), intent(in) :: units
228     if (allocated(self%vectorial3d%field)) then
229         stop '[generic_field_class::initialize]: vectorial 3D field already allocated'
230     else
231         call self%vectorial3d%initialize(name, units, 3, field)
232     end if
```

Here is the caller graph for this function:



6.9.2.12 initv4d()

```
subroutine field_types_mod::initv4d (
            class(generic_field_class), intent(inout) self,
            type(string), intent(in) name,
            type(string), intent(in) units,
            type(vector), dimension(:,:,:,:), intent(in) field )  [private]
```

Method that allocates and initializes a vectorial 4D field in a generic field.

Author

Ricardo Birjukovs Canelas - MARETEC

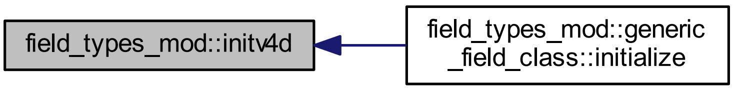
Parameters

in	*self,name,units,field*	

Definition at line 242 of file fields_types.f90.

```
242     class(generic_field_class), intent(inout) :: self
243     type(vector), intent(in), dimension(:,:,:,:) :: field
244     type(string), intent(in) :: name
245     type(string), intent(in) :: units
246     if (allocated(self%vectorial4d%field)) then
247         stop '[generic_field_class::initialize]: vectorial 4D field already allocated'
248     else
249         call self%vectorial4d%initialize(name, units, 4, field)
250     end if
```

Here is the caller graph for this function:



6.9.2.13 initvectorial2dfield()

```
subroutine field_types_mod::initvectorial2dfield (
            class(vectorial2d_field_class), intent(inout) self,
            type(string), intent(in) name,
            type(string), intent(in) units,
            integer, intent(in) dim,
            type(vector), dimension(:,:), intent(in) field )  [private]
```

Method that initializes a vectorial 2D field.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,name,units,dim,field*	

Definition at line 324 of file fields_types.f90.

```
324     class(vectorial2d_field_class), intent(inout) :: self
```

```
325     type(vector), intent(in), dimension(:,:) :: field
326     type(string), intent(in) :: name
327     type(string), intent(in) :: units
328     integer, intent(in) :: dim
329     call self%setFieldMetadata(name, units, dim)
330     allocate(self%field, source = field)
```

6.9.2.14 initvectorial3dfield()

```
subroutine field_types_mod::initvectorial3dfield (
            class(vectorial3d_field_class), intent(inout) self,
            type(string), intent(in) name,
            type(string), intent(in) units,
            integer, intent(in) dim,
            type(vector), dimension(:,:,:), intent(in) field )  [private]
```

Method that initializes a vectorial 3D field.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,name,units,dim,field*	

Definition at line 340 of file fields_types.f90.

```
340     class(vectorial3d_field_class), intent(inout) :: self
341     type(vector), intent(in), dimension(:,:,:) :: field
342     type(string), intent(in) :: name
343     type(string), intent(in) :: units
344     integer, intent(in) :: dim
345     call self%setFieldMetadata(name, units, dim)
346     allocate(self%field, source = field)
```

6.9.2.15 initvectorial4dfield()

```
subroutine field_types_mod::initvectorial4dfield (
            class(vectorial4d_field_class), intent(inout) self,
            type(string), intent(in) name,
            type(string), intent(in) units,
            integer, intent(in) dim,
            type(vector), dimension(:,:,:,:), intent(in) field )  [private]
```

Method that initializes a vectorial 4D field.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,name,units,dim,field*	

Definition at line 356 of file fields_types.f90.

```
356     class(vectorial4d_field_class), intent(inout) :: self
357     type(vector), intent(in), dimension(:,:,:,:) :: field
358     type(string), intent(in) :: name
359     type(string), intent(in) :: units
360     integer, intent(in) :: dim
361     call self%setFieldMetadata(name, units, dim)
362     allocate(self%field, source = field)
```

6.9.2.16 printfield()

```
subroutine field_types_mod::printfield (
            class(field_class), intent(in) self )  [private]
```

Method that prints the field information.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 434 of file fields_types.f90.

```
434     class(field_class), intent(in) :: self
435     type(string) :: outext, t(2)
436     t(1) = self%dim
437     t(2) = self%getFieldType()
438     outext = t(2)//' field['//self%name//'] has dimensionality '//t(1)//' and is in '//self%units
439     call log%put(outext,.false.)
```

6.9.2.17 printgenericfield()

```
subroutine field_types_mod::printgenericfield (
            class(generic_field_class), intent(in) self )  [private]
```

Method that prints the generic field information.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 388 of file fields_types.f90.

```
388     class(generic_field_class), intent(in) :: self
389     if (allocated(self%scalar1d%field)) call self%scalar1d%print()
390     if (allocated(self%scalar2d%field)) call self%scalar2d%print()
391     if (allocated(self%scalar3d%field)) call self%scalar3d%print()
392     if (allocated(self%scalar4d%field)) call self%scalar4d%print()
393     if (allocated(self%vectorial2d%field)) call self%vectorial2d%print()
394     if (allocated(self%vectorial3d%field)) call self%vectorial3d%print()
395     if (allocated(self%vectorial4d%field)) call self%vectorial4d%print()
```

6.9.2.18 setfieldmetadata()

```
subroutine field_types_mod::setfieldmetadata (
            class(field_class), intent(inout) self,
            type(string), intent(in) name,
            type(string), intent(in) units,
            integer, intent(in) dim )  [private]
```

Method that initializes a base field object by filling metadata.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,name,units,dim*	

Definition at line 373 of file fields_types.f90.

```
373     class(field_class), intent(inout) :: self
374     type(string), intent(in) :: name
375     type(string), intent(in) :: units
376     integer, intent(in) :: dim
377     self%name = name
378     self%units = units
379     self%dim = dim
```

6.9.2.19 test()

```
subroutine field_types_mod::test (
            class(generic_field_class), intent(inout) self )  [private]
```

A class 'unit' test for the generic_field_class.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 404 of file fields_types.f90.

```
404     class(generic_field_class), intent(inout) :: self
405     type(generic_field_class) :: gfield1, gfield2, gfield3
406     real(prec), allocatable, dimension(:) :: field1
407     real(prec), allocatable, dimension(:,:) :: field2
408     type(vector), allocatable, dimension(:,:,:) :: field3
409     type(string) :: name1, name2, name3
410     type(string) :: units1, units2, units3
411     allocate(field1(50))
412     allocate(field2(20,60))
413     allocate(field3(2,3,4))
414     name1 = 'testfield1d'
415     name2 = 'testfield2d'
416     name3 = 'testfield3d'
417     units1 = 'm/s'
418     units2 = 'km'
419     units3 = 'ms-1'
420     call gfield1%initialize(name1, units1, field1)
421     call gfield2%initialize(name2, units2, field2)
422     call gfield3%initialize(name3, units3, field3)
423     call gfield1%print()
424     call gfield2%print()
425     call gfield3%print()
```

6.10 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, dp)

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

  method to get the list of points that fill a given geometry
- type(vector) function getcenter (self, shapetype)

  method to get the baricenter of a given geometry
- type(vector) function, dimension(:), allocatable getpoints (self, shapetype)

  method that returns the points defining a given geometry
- integer function getnumpoints (self, shapetype)

  method the points defining 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

6.10.1 Detailed Description

Module that defines geometry classes and related methods.

Author

Ricardo Birjukovs Canelas

6.10.2 Function/Subroutine Documentation

6.10.2.1 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 79 of file geometry.f90.

```
79     implicit none
80     class(geometry_class), intent(inout) :: self
81     allocate(self%list(4))
82     self%list(1) ='point'
83     self%list(2) ='line'
84     self%list(3) ='box'
85     self%list(4) ='sphere'
```

6.10.2.2 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

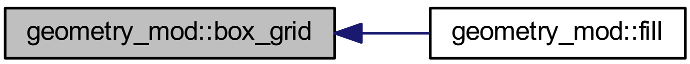
Parameters

in	*dp,size,np,ptlist*	

Definition at line 397 of file geometry.f90.

```
397     implicit none
398     real(prec), intent(in) :: dp
399     type(vector), intent(in) :: size
400     integer, intent(in)::  np
401     type(vector), intent(out) :: ptlist(np)
402     integer :: i, j, k, p
403     p=0
404     do i=1, int(size%x/dp)+1
405         do j=1, int(size%y/dp)+1
406             do k=1, int(size%z/dp)+1
407                 p=p+1
408                 ptlist(p) = dp*(ex*(i-1) + ey*(j-1) + ez*(k-1))
409             end do
410         end do
411     end do
412     if (np == 1) then !Just the origin
413         ptlist(1)= 0*ex + 0*ey +0*ez
414     end if
```

Here is the caller graph for this function:



6.10.2.3 fill()

```
subroutine geometry_mod::fill (
            class(geometry_class), intent(in) self,
            class(shape) shapetype,
            real(prec), intent(in) dp,
            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

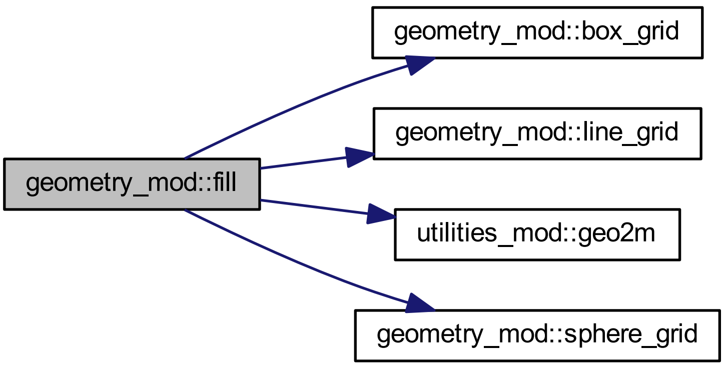
Parameters

in	*self,shapetype,dp,fillsize,ptlist*	

Definition at line 147 of file geometry.f90.

```
147     implicit none
148     class(geometry_class), intent(in) :: self
149     class(shape) :: shapetype
150     real(prec), intent(in) :: dp
151     integer, intent(in) :: fillsize
152     type(vector), intent(out) :: ptlist(fillsize)
153     type(vector) :: temp
154     type(string) :: outext
155     select type (shapetype)
156     type is (shape)
157     class is (box)
158         call box_grid(dp, shapetype%size, fillsize, ptlist)
159     class is (point)
160         ptlist(1)=0
161     class is (line)
162         call line_grid(dp, geo2m(shapetype%last-shapetype%pt, shapetype%pt%y), fillsize, ptlist)
163     class is (sphere)
164         call sphere_grid(dp, shapetype%radius, fillsize, ptlist)
165         class default
166         outext='[geometry::fill] : unexpected type for geometry object, stoping'
167         call log%put(outext)
168         stop
169     end select
```

Here is the call graph for this function:



6.10.2.4 fillsize()

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

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

Author

Ricardo Birjukovs Canelas - MARETEC

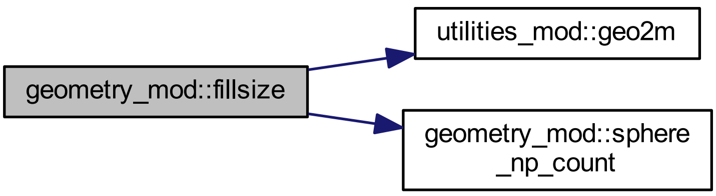
Parameters

in	*self,shapetype,dp*	

Definition at line 114 of file geometry.f90.

```
114     implicit none
115     class(geometry_class), intent(in) :: self
116     class(shape), intent(in) :: shapetype
117     real(prec), intent(in) :: dp
118     integer :: fillsize
119     type(vector) :: temp
120     type(string) :: outext
121     select type (shapetype)
122     type is (shape)
123     class is (box)
124         fillsize = max((int(shapetype%size%x/dp)+1)*(int(shapetype%size%y/dp)+1)*(int(shapetype%size%z/dp)+1),1)
125     class is (point)
126         fillsize = 1
127     class is (line)
128         temp = shapetype%pt - shapetype%last
129         temp = geo2m(temp, shapetype%pt%y)
130         fillsize = max(int(temp%normL2()/dp),1)
131     class is (sphere)
132         fillsize = sphere_np_count(dp, shapetype%radius)
133         class default
134         outext='[geometry::fillsize] : unexpected type for geometry object, stoping'
135         call log%put(outext)
136         stop
137     end select
```

Here is the call graph for this function:



6.10.2.5 getcenter()

```
type(vector) function geometry_mod::getcenter (
            class(geometry_class), intent(in) self,
            class(shape), intent(in) shapetype )  [private]
```

method to get the baricenter of a given geometry

Author

Ricardo Birjukovs Canelas - MARETEC

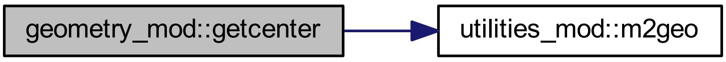
Parameters

in	*self,shapetype*	

Definition at line 179 of file geometry.f90.

```
179     implicit none
180     class(geometry_class), intent(in) :: self
181     class(shape), intent(in) :: shapetype
182     type(vector) :: center
183     type(string) :: outext
184     select type (shapetype)
185     type is (shape)
186     class is (box)
187         center = shapetype%pt + m2geo(shapetype%size, shapetype%pt%y)/2.0
188     class is (point)
189         center = shapetype%pt
190     class is (line)
191         center = shapetype%pt + shapetype%last/2.0
192     class is (sphere)
193         center = shapetype%pt
194         class default
195         outext='[geometry::getCenter] : unexpected type for geometry object, stoping'
196         call log%put(outext)
197         stop
198     end select
```

Here is the call graph for this function:



6.10.2.6 getnumpoints()

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

method the points defining a given geometry

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,shapetype*	

Definition at line 256 of file geometry.f90.

```
256     class(geometry_class), intent(in) :: self
257     class(shape), intent(in) :: shapetype
258     integer :: n
259     type(string) :: outext
260     select type (shapetype)
261     type is (shape)
262     class is (box)
263         n=8
264     class is (point)
265         n=1
266     class is (line)
267         n=2
268     class is (sphere)
269         n=1
270         class default
271         outext='[geometry::getnumPoints] : unexpected type for geometry object, stoping'
272         call log%put(outext)
273         stop
274     end select
```

6.10.2.7 getpoints()

type(vector) function, dimension(:), allocatable geometry_mod::getpoints (
class(geometry_class), intent(in) *self,*
class(shape), intent(in) *shapetype*) [private]

method that returns the points defining a given geometry

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,shapetype*	

Definition at line 208 of file geometry.f90.

```
208     class(geometry_class), intent(in) :: self
209     class(shape), intent(in) :: shapetype
210     type(vector), allocatable :: pts(:)
211     type(string) :: outext
212     integer :: n
213     type(vector) :: temp
214     select type (shapetype)
215     type is (shape)
216     class is (box)
217         n=8
218         allocate(pts(n))
219         temp = shapetype%size
220         !temp = m2geo(shapetype%size, shapetype%pt%y)
221         pts(1) = shapetype%pt
222         pts(2) = shapetype%pt + temp%y*ey
223         pts(3) = pts(2) + temp%z*ez
224         pts(4) = shapetype%pt + temp%z*ez
225         pts(5) = shapetype%pt + temp%x*ex
226         pts(6) = pts(5) + temp%y*ey
227         pts(7) = shapetype%pt + temp
228         pts(8) = pts(5) + temp%z*ez
229     class is (point)
230         n=1
231         allocate(pts(n))
```

```
        pts(1) = shapetype%pt
    class is (line)
        n=2
        allocate(pts(n))
        pts(1) = shapetype%pt
        pts(2) = shapetype%last
    class is (sphere)
        n=1
        allocate(pts(n))
        pts(1) = shapetype%pt
        class default
        outext='[geometry::getPoints] : unexpected type for geometry object, stoping'
        call log%put(outext)
        stop
    end select
```

6.10.2.8 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	*self,geomname*	

Definition at line 95 of file geometry.f90.

```
95      implicit none
96      class(geometry_class), intent(in) :: self
97      type(string), intent(in) :: geomname
98      integer :: i
99      tf = .false.
100      do i=1, size(self%list)
101          if (geomname == self%list(i)) then
102              tf = .true.
103          endif
104      enddo
```

6.10.2.9 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

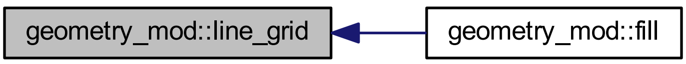
Parameters

in	*dp,dist,np,ptlist*	

Definition at line 425 of file geometry.f90.

```
425     implicit none
426     real(prec), intent(in) :: dp
427     type(vector), intent(in) :: dist
428     integer, intent(in)::  np
429     type(vector), intent(out) :: ptlist(np)
430     integer :: i, j, k, p
431 
432     do p=1, np
433         ptlist(p) = dp/np*(dist*(p-1))
434     end do
435     if (np == 1) then !Just the origin
436         ptlist(1)= 0*ex + 0*ey +0*ez
437     end if
```

Here is the caller graph for this function:



6.10.2.10 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	*self,shapetype*	

Definition at line 284 of file geometry.f90.

```
284     implicit none
285     class(geometry_class), intent(in) :: self
286     class(shape) :: shapetype
287 
288     type(vector) :: temp(2)
289     type(string) :: temp_str(6)
290     type(string) :: outext
291 
292     temp_str(1) = shapetype%pt%x
293     temp_str(2) = shapetype%pt%y
294     temp_str(3) = shapetype%pt%z
295     select type (shapetype)
296     type is (shape)
297     class is (box)
298         temp_str(4) = shapetype%size%x
299         temp_str(5) = shapetype%size%y
300         temp_str(6) = shapetype%size%z
301         outext='      Box at '//temp_str(1)//' '//temp_str(2)//' '//temp_str(3)//new_line('a')//&
302             '       with '//temp_str(4)//' X '//temp_str(5)//' X '//temp_str(6)
303     class is (point)
304         outext='      Point at '//temp_str(1)//' '//temp_str(2)//' '//temp_str(3)
305     class is (line)
306         temp_str(4) = shapetype%last%x
307         temp_str(5) = shapetype%last%y
308         temp_str(6) = shapetype%last%z
309         outext='      Line from '//temp_str(1)//' '//temp_str(2)//' '//temp_str(3)//new_line('a')//&
310             '       to '//temp_str(4)//' X '//temp_str(5)//' X '//temp_str(6)
311     class is (sphere)
312         temp_str(4) = shapetype%radius
313         outext='      Sphere at '//temp_str(1)//' '//temp_str(2)//' '//temp_str(3)//new_line('a')//&
314             '       with radius '//temp_str(4)
315         class default
316         outext='[geometry::print] : unexpected type for geometry object, stoping'
317         call log%put(outext)
318         stop
319     end select
320     call log%put(outext,.false.)
321 
```

6.10.2.11 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

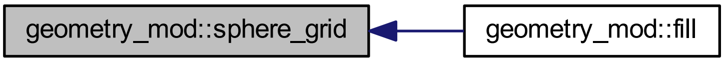
in	*dp,r,np,ptlist*	

Definition at line 363 of file geometry.f90.

```
363     implicit none
364     real(prec), intent(in) :: dp
```

```
    real(prec), intent(in) :: r
    integer, intent(in)::  np
    type(vector), intent(out) :: ptlist(np)
    integer :: i, j, k, p, n
    type(vector) :: pts
    n=int(3*r/dp)
    p=0
    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
                    p=p+1
                    ptlist(p)=pts
                end if
            end do
        end do
    end do
    if (np == 1) then !Just the center point
        ptlist(1)= 0*ex + 0*ey +0*ez
    end if

```

Here is the caller graph for this function:



6.10.2.12 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

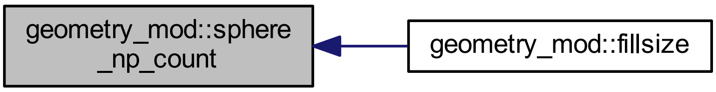
in	*dp,r*	

Definition at line 332 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:



6.10.3 Variable Documentation

6.10.3.1 geometry

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

Definition at line 65 of file geometry.f90.

```
65      type(geometry_class) :: Geometry
```

6.11 link_mod Module Reference

Module that defines a link based on an unlimited polymorphic container class.

Data Types

- interface link

Functions/Subroutines

- class(*) function, pointer getvalue (this)

  Method that returns a pointer to the values stored in the container in this link.
- class(link) function, pointer nextlink (this)

  Method that returns a pointer to the next link in a list.
- class(link) function, pointer previouslink (this)

  Method that returns a pointer to the previous link in a list.
- subroutine setnextlink (this, next)

  Method to set the next link in a list.
- subroutine setpreviouslink (this, prev)

  Method to set the previous link in a list.
- subroutine removelink (this)

  Method to remove a link in a list.
- class(link) function, pointer constructor (to_store, prev, next, key)

  Link constructor, can be used with the 'link' name since it was defined as such in an interface declaration.

6.11.1 Detailed Description

Module that defines a link based on an unlimited polymorphic container class.

Author

Ricardo Birjukovs Canelas

6.11.2 Function/Subroutine Documentation

6.11.2.1 constructor()

```
class(link) function, pointer link_mod::constructor (
            class(*), intent(in) to_store,
            class(link), intent(in), pointer prev,
            class(link), intent(in), pointer next,
            integer, intent(in), optional key )  [private]
```

Link constructor, can be used with the 'link' name since it was defined as such in an interface declaration.

Author

Ricardo Birjukovs Canelas - MARETEC

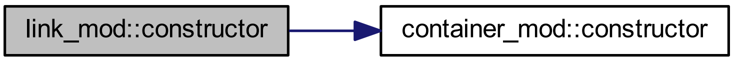
Parameters

[to_store,prev,next]	

Definition at line 134 of file link.f90.

```
134     class(link), pointer :: constructor
135     class(*), intent(in) :: to_store
136     class(link), pointer, intent(in) :: prev
137     class(link), pointer, intent(in) :: next
138     integer, intent(in), optional :: key
139     allocate(constructor)
140     call constructor%setPreviousLink(prev)
141     call constructor%setNextLink(next)
142     call constructor%storeContent(to_store)
143     if (present(key)) then
144         constructor%key = key
145     end if
```

Here is the call graph for this function:



6.11.2.2 getvalue()

```
class(*) function, pointer link_mod::getvalue (
            class(link) this )  [private]
```

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

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 67 of file link.f90.

```
67     class(link) :: this
68     class(*), pointer :: getValue
69     getvalue => this%getContent()
```

6.11.2.3 nextlink()

```
class(link) function, pointer link_mod::nextlink (
            class(link) this )  [private]
```

Method that returns a pointer to the next link in a list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 78 of file link.f90.

```
78     class(link) :: this
79     class(link), pointer :: nextLink
80     nextlink => this%next
```

6.11.2.4 previouslink()

class(link) function, pointer link_mod::previouslink (
    class(link) *this*) [private]

Method that returns a pointer to the previous link in a list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 89 of file link.f90.

```
89      class(link) :: this
90      class(link), pointer :: previousLink
91      previouslink => this%previous
```

6.11.2.5 removelink()

subroutine link_mod::removelink (
    class(link), intent(inout) *this*) [private]

Method to remove a link in a list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 122 of file link.f90.

```
122      class(link), intent(inout) :: this
123      call this%deleteContent()
```

6.11.2.6 setnextlink()

subroutine link_mod::setnextlink (
    class(link) *this*,
    class(link), pointer *next*) [private]

Method to set the next link in a list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 100 of file link.f90.

```
100      class(link) :: this
101      class(link), pointer :: next
102      this%next => next
```

6.11.2.7 setpreviouslink()

```
subroutine link_mod::setpreviouslink (
            class(link) this,
            class(link), pointer prev )  [private]
```

Method to set the previous link in a list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 111 of file link.f90.

```
class(link) :: this
class(link), pointer :: prev
this%previous => prev
```

6.12 simulation_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

6.12.1 Detailed Description

Module to print version, licence, preambles.

Author

Ricardo Birjukovs Canelas

6.12.2 Function/Subroutine Documentation

6.12.2.1 printlicpreamble()

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

Public licence and preamble printer routine.

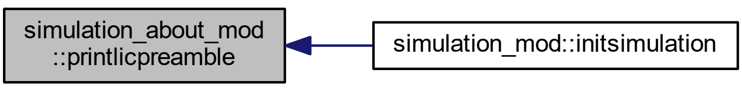
Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 44 of file simulation_about.f90.

```
44     implicit none
45     type(string) :: outext
46
47     version  ="v0.2"
48     author   ="R. Birjukovs Canelas"
49     date     ="24-08-2018"
50
51     outext = ' __  __  ___  _   _ ___ ____  _                                  _             '//
      new_line('a')//&
52              ' |  \/  |/ _ \| | | |_ _|  _ \| |    __ _  __ _ _ __ __ _ _ __   __ _(_) __ _ _ __  '//
      new_line('a')//&
53              ' | |\/| | | | | |_| || || | | | |   / _` |/ _` | __/ _` |  _ \ / _` | |/ _` |  _ \ '//
      new_line('a')//&
54              ' | |  | | |_| |  _  || || |_| | |__| (_| | (_| | | | (_| | | | | (_| | | (_| | | | |'//
      new_line('a')//&
55              ' |_|  |_|\___/|_| |_|___|____/|_____\__,_|\__, |_|  \__,_|_| |_|\__, |_|\__,_|_| |_|'//
      new_line('a')//&
56              '                                          |___/                 |___/               '//
      new_line('a')//&
57
58          '  <MOHIDLagrangian> Copyright (C) 2018 by'//new_line('a')//&
59          '  R. Birjukovs Canelas, R. Neves, F. Campuzano, H. de Pablo Lenonardo'//new_line('a')//&
60          ''//new_line('a')//&
61          '  MARETEC - Research Centre for Marine, Environment and Technology'//new_line('a')//&
62          ''//new_line('a')//&
63          '  MOHIDLagrangian is free software: you can redistribute it and/or'//new_line('a')//&
64          '  modify it under the terms of the GNU General Public License as'//new_line('a')//&
65          '  published by the Free Software Foundation, either version 3 of'//new_line('a')//&
66          '  the License, or (at your option) any later version.'//new_line('a')//&
67          ''//new_line('a')//&
68          '  MOHIDLagrangian is distributed WITHOUT ANY WARRANTY; without even'//new_line('a')//&
69          '  the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR'//new_line('a')//&
70          '  PURPOSE. See the GNU General Public License for more details.'//new_line('a')//&
71          ''//new_line('a')//&
72          '  You should have received a copy of the GNU General Public License,'//new_line('a')//&
73          '  along with MOHIDLagrangian. If not, see <http://www.gnu.org/licenses/>.,'//new_line('a')//&
74          ''//new_line('a')//&
75          ''//new_line('a')//&
76          'MOHIDLagrangian '//version//' ('//author//') ('//date//')'//new_line('a')//&
77          '======================================================================'
78
79     !call Log%put(outext,.false.)
80     call log%put(outext,.false.)
81
```

Here is the caller graph for this function:



6.12.3 Variable Documentation

6.12.3.1 author

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

Definition at line 31 of file simulation_about.f90.

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

6.12.3.2 date

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

Definition at line 32 of file simulation_about.f90.

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

6.12.3.3 version

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

Definition at line 30 of file simulation_about.f90.

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

6.13 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 sim_t

  Simulation related counters and others.
- type simdefs_t

  Simulation definitions class.
- type src_parm_t

  Lists for Source parameters.

Functions/Subroutines

- subroutine setdefaults (self, outpath)

    Globals default setting routine.
- subroutine increment_numtracer (self)

    Increments Tracer count. This routine MUST be ATOMIC.
- integer function getnumtracer (self)

    Returns a new ID for a Tracer.
- subroutine increment_numdt (self)

    incrementing time step count.
- integer function getnumdt (self)

    Returns the number of time steps.
- subroutine increment_numoutfile (self)

    incrementing output file count.
- integer function getnumoutfile (self)

    Returns the number of output files written.
- subroutine buildlists (self)

    Method to build the parameters list of the Sources.
- 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 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

6.13.1 Detailed Description

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

Author

Ricardo Birjukovs Canelas

6.13.2 Function/Subroutine Documentation

6.13.2.1 buildlists()

```
subroutine simulation_globals_mod::buildlists (
            class(src_parm_t), intent(inout) self )  [private]
```

Method to build the parameters list of the Sources.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 270 of file simulation_globals.f90.

```
implicit none
class(src_parm_t), intent(inout) :: self
allocate(self%baselist(5))
self%baselist(1) = 'particulate'
self%baselist(2) = 'density'
self%baselist(3) = 'radius'
self%baselist(4) = 'condition'
self%baselist(5) = 'degradation_rate'
allocate(self%particulatelist(2))
self%particulatelist(1) = 'intitial_concentration'
self%particulatelist(2) = 'particle_radius'
```

6.13.2.2 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 354 of file simulation_globals.f90.

```
354     implicit none
355     class(parameters_t), intent(inout) :: self
356     type(string) :: outext
357     type(datetime) :: temp
358     type(timedelta) :: simtime
359
360     if ( any(self%IntegratorIndexes == self%Integrator)) then
361     else
362         outext = '[Globals::parameters::check]: Integrator not recognized, stoping'
363         call log%put(outext)
364         stop
365     end if
366     if ( any(self%OutputFormatIndexes == self%OutputFormat)) then
367         if (self%OutputFormat == 1) then
368             outext = '[Globals::parameters::check]: NetCDF is not implemented yet, try something nicer like VTK, stoping'
369             call log%put(outext)
370             stop
371         end if
372     else
373         outext = '[Globals::parameters::check]: OutputFormat not recognized, stoping'
374         call log%put(outext)
375         stop
376     end if
377     temp = datetime() !default initialization
378     !add new parameters to this search
379     if (self%TimeOut==mv) then
380         outext = '[Globals::parameters::check]: sampling rate parameter (TimeOut) is not set, stoping'
381         call log%put(outext)
382         stop
383     elseif (self%StartTime==temp) then
384         outext = '[Globals::parameters::check]: start time parameter (StartTime) is not set or invalid, stoping'
385         call log%put(outext)
386         stop
387     elseif (self%EndTime==temp) then
388         outext = '[Globals::parameters::check]: end time parameter (EndTime) is not set or invalid, stoping'
389         call log%put(outext)
390         stop
391     end if
392     !Build timemax from the difference between start and end time
393     simtime = self%EndTime - self%StartTime
394     self%TimeMax = simtime%total_seconds()
```

6.13.2.3 getnumdt()

```
integer function simulation_globals_mod::getnumdt (
            class(sim_t), intent(inout) self )  [private]
```

Returns the number of time steps.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 237 of file simulation_globals.f90.

```
237     implicit none
238     class(sim_t), intent(inout) :: self
239     getnumdt = self%numdt
```

6.13.2.4 getnumoutfile()

```
integer function simulation_globals_mod::getnumoutfile (
            class(sim_t), intent(inout) self )  [private]
```

Returns the number of output files written.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 259 of file simulation_globals.f90.

```
implicit none
class(sim_t), intent(inout) :: self
getnumoutfile = self%numoutfile
```

6.13.2.5 getnumtracer()

```
integer function simulation_globals_mod::getnumtracer (
            class(sim_t), intent(inout) self )  [private]
```

Returns a new ID for a Tracer.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 214 of file simulation_globals.f90.

```
implicit none
class(sim_t), intent(inout) :: self
call self%increment_numTracer()
getnumtracer = self%numTracer
```

6.13.2.6 increment_numdt()

```
subroutine simulation_globals_mod::increment_numdt (
            class(sim_t), intent(inout) self )  [private]
```

incrementing time step count.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 226 of file simulation_globals.f90.

```
implicit none
class(sim_t), intent(inout) :: self
self%numdt = self%numdt + 1
```

6.13.2.7 increment_numoutfile()

```
subroutine simulation_globals_mod::increment_numoutfile (
            class(sim_t), intent(inout) self )  [private]
```

incrementing output file count.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 248 of file simulation_globals.f90.

```
248     implicit none
249     class(sim_t), intent(inout) :: self
250     self%numoutfile = self%numoutfile + 1
```

6.13.2.8 increment_numtracer()

```
subroutine simulation_globals_mod::increment_numtracer (
            class(sim_t), intent(inout) self )  [private]
```

Increments Tracer count. This routine MUST be ATOMIC.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 202 of file simulation_globals.f90.

```
202     implicit none
203     class(sim_t), intent(inout) :: self
204     !ATOMIC pragma here please
205     self%numTracer = self%numTracer + 1
```

6.13.2.9 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 493 of file simulation_globals.f90.

```
493     implicit none
494     class(constants_t), intent(in) :: self
495     type(string) :: outext
496     type(string) :: temp_str(3)
497
498     temp_str(1)=self%Gravity%x
499     temp_str(2)=self%Gravity%y
500     temp_str(3)=self%Gravity%z
501     outext = '      Gravity is '//new_line('a')//&
502         '        '//temp_str(1)//' '//temp_str(2)//' '//temp_str(3)//new_line('a')
503     temp_str(1)=self%Z0
504     outext = outext//'       Z0 = '//temp_str(1)//' m'//new_line('a')
505     temp_str(1)=self%Rho_ref
506     outext = outext//'       Rho_ref = '//temp_str(1)//' kg/m^3'
507
508     call log%put(outext,.false.)
```

6.13.2.10 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 598 of file simulation_globals.f90.

```
implicit none
class(simdefs_t), intent(in) :: self
type(string) :: outext
type(string) :: temp_str(3)

temp_str(1)=self%Dp
outext = '      Initial resolution is '//temp_str(1)//' m'//new_line('a')
temp_str(1)=self%dt
outext = '      Timestep is '//temp_str(1)//' s'//new_line('a')
temp_str(1)=self%Pointmin%x
temp_str(2)=self%Pointmin%y
temp_str(3)=self%Pointmin%z
outext = outext//'       Pointmin (BB) is '//new_line('a')//&
    '       '//temp_str(1)//' '//temp_str(2)//' '//temp_str(3)//new_line('a')
temp_str(1)=self%Pointmax%x
temp_str(2)=self%Pointmax%y
temp_str(3)=self%Pointmax%z
outext = outext//'       Pointmax (BB) is '//new_line('a')//&
    '       '//temp_str(1)//' '//temp_str(2)//' '//temp_str(3)//new_line('a')
if (self%autoblocksize) then
    outext = outext//'       Blocks are automatically sized'
else
    temp_str(1)=self%blocksize%x
    temp_str(2)=self%blocksize%y
    outext = outext//'       Blocks are sized '//new_line('a')//&
        '       '//temp_str(1)//' X '//temp_str(2)
end if
call log%put(outext,.false.)
```

6.13.2.11 printsimparameters()

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

Parameter printing method.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 403 of file simulation_globals.f90.

```
implicit none
class(parameters_t), intent(inout) :: self
type(string) :: outext
type(string) :: temp_str
character(len=23) :: temp_char
outext = '      Integrator scheme is '//self%IntegratorNames(self%Integrator)//new_line('a')
temp_str=self%CFL
outext = outext//'       CFL = '//temp_str//new_line('a')
temp_str=self%WarmUpTime
outext = outext//'       WarmUpTime = '//temp_str//' s'//new_line('a')
temp_str=self%TimeOut
outext = outext//'       TimeOut = '//temp_str//' Hz'//new_line('a')
temp_char = self%StartTime%isoformat(' ')
temp_str = temp_char
outext = outext//'       StartTime = '//temp_str//new_line('a')
temp_char = self%EndTime%isoformat(' ')
temp_str = temp_char
outext = outext//'       EndTime   = '//temp_str//new_line('a')
temp_str=self%TimeMax
outext = outext//'       Simulation will run for '//temp_str//' s'//new_line('a')
outext = outext//'       Output file format is '//self%OutputFormatNames(self%OutputFormat)
call log%put(outext,.false.)
```

6.13.2.12 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	*self,bsize*	

Definition at line 583 of file simulation_globals.f90.

```
implicit none
class(simdefs_t), intent(inout) :: self
type(vector) :: bsize
integer :: sizem
self%blocksize = bsize
sizem = sizeof(bsize)
call simmemory%adddef(sizem)
```

6.13.2.13 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	*self,point_,coords*	

Definition at line 562 of file simulation_globals.f90.

```
implicit none
class(simdefs_t), intent(inout) :: self
type(string), intent(in) :: point_
type(vector) :: coords
integer :: sizem
if (point_%chars() == "pointmin") then
    self%Pointmin= coords
elseif (point_%chars() == "pointmax") then
    self%Pointmax= coords
endif
sizem=sizeof(coords)
call simmemory%adddef(sizem)
```

6.13.2.14 setdefaults()

```
subroutine simulation_globals_mod::setdefaults (
            class(globals_class), intent(inout) self,
            type(string), intent(in), optional outpath )  [private]
```

Globals default setting routine.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,outpath*	

Definition at line 137 of file simulation_globals.f90.

```
implicit none
class(globals_class), intent(inout) :: self
integer :: sizem
type(string), optional, intent(in) :: outpath
!parameters
self%Parameters%Integrator = 1
self%Parameters%IntegratorIndexes = [1,2,3]
self%Parameters%IntegratorNames(1) = 'Verlet'
self%Parameters%IntegratorNames(2) = 'Symplectic'
self%Parameters%IntegratorNames(3) = 'Runge-Kuta 4'
self%Parameters%CFL = 0.5
self%Parameters%WarmUpTime = 0.0
self%Parameters%TimeOut = mv
self%Parameters%TimeOut = mv
self%Parameters%StartTime = datetime()
self%Parameters%EndTime = datetime()
```

```
    self%Parameters%OutputFormat = 2
    self%Parameters%OutputFormatIndexes = [1,2]
    !self%Parameters%OutputFormatNames = ['NetCDF','VTK'] !This is not acceptable because FORTRAN
    self%Parameters%OutputFormatNames(1) = 'NetCDF'
    self%Parameters%OutputFormatNames(2) = 'VTK'
    !Simulation definitions
    self%SimDefs%autoblocksize =.true.
    self%SimDefs%blocksize = 0.0
    self%SimDefs%numblocksx = mv
    self%SimDefs%numblocksy = mv
    self%SimDefs%numblocks = 16  !placeholder number, should be numThreads or numProcesses or computed by user dimensions
    self%SimDefs%Dp = mv
    self%SimDefs%dt = mv
    self%SimDefs%Pointmin = 0.0
    self%SimDefs%Pointmax = 0.0
    !simulation constants
    self%Constants%Gravity= 0.0*ex + 0.0*ey -9.81*ez
    self%Constants%Z0 = 0.0
    self%Constants%Rho_ref = 1000.0
    !filenames
    self%Names%mainxmlfilename = 'not_set'
    self%Names%propsxmlfilename = 'not_set'
    self%Names%tempfilename = 'not_set'
    if (present(outpath)) then
        self%Names%outpath = outpath
    else
        self%Names%outpath = 'not_set'
    end if
    self%Names%casename = 'not_set'
    !global time
    self%SimTime = 0.0
    !global counters
    self%Sim%numdt = 0
    self%Sim%numoutfile = 0
    self%Sim%numTracer = 0
    !Source parameters list
    call self%SrcProp%buildlists()

    sizem=sizeof(self)
    call simmemory%adddef(sizem)

```

6.13.2.15 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	*self,read_dp*	

Definition at line 518 of file simulation_globals.f90.

```
518     implicit none
519     class(simdefs_t), intent(inout) :: self
520     type(string), intent(in) :: read_dp
521     type(string) :: outext
522     integer :: sizem
```

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

6.13.2.16 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	*self,read←_dt*	

Definition at line 540 of file simulation_globals.f90.

```
implicit none
class(simdefs_t), intent(inout) :: self
type(string), intent(in) :: read_dt
type(string) :: outext
integer :: sizem
self%dt=read_dt%to_number(kind=1._r4p)
if (self%dt.le.0.0) then
    outext='dt must be positive and non-zero, stopping'
    call log%put(outext)
    stop
endif
sizem = sizeof(self%dt)
call simmemory%adddef(sizem)
```

6.13.2.17 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	*self,grav*	

Definition at line 434 of file simulation_globals.f90.

```
434     implicit none
435     class(constants_t), intent(inout) :: self
436     type(vector), intent(in) :: grav
437     integer :: sizem
438     type(string) :: outext
439     self%Gravity= grav
440     if (grav%x==mv) then !Gravity was not read, setting default
441         self%Gravity= -9.81*ez
442         outext = '        Gravity not specified, setting to default value = (0,0,-9.81)'
443         call log%put(outext,.false.)
444     endif
445     sizem=sizeof(self%Gravity)
446     call simmemory%adddef(sizem)
```

6.13.2.18 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	*self,parmkey,parmvalue*	

Definition at line 290 of file simulation_globals.f90.

```
290     implicit none
291     class(parameters_t), intent(inout) :: self
292     type(string), intent(in) :: parmkey
293     type(string), intent(in) :: parmvalue
294     type(string), allocatable :: dc(:)
295     integer :: i, date(6)
296     integer :: sizem
297     !add new parameters to this search
298     if (parmkey%chars()=="Integrator") then
299         self%Integrator=parmvalue%to_number(kind=1_i1p)
300         sizem=sizeof(self%Integrator)
301     elseif(parmkey%chars()=="CFL") then
302         self%CFL=parmvalue%to_number(kind=1._r4p)
303         sizem=sizeof(self%CFL)
304     elseif(parmkey%chars()=="WarmUpTime") then
305         self%WarmUpTime=parmvalue%to_number(kind=1._r4p)
306         sizem=sizeof(self%WarmUpTime)
307     elseif(parmkey%chars()=="TimeOut") then
308         self%TimeOut=parmvalue%to_number(kind=1._r4p)
309         sizem=sizeof(self%TimeOut)
```

```
    elseif(parmkey%chars()=="StartTime") then
        call parmvalue%split(tokens=dc, sep=' ')
        if (size(dc) == 6) then
            do i=1, size(dc)
                date(i) = dc(i)%to_number(kind=1._r4p)
            end do
            self%StartTime = datetime(date(1),date(2),date(3),date(4),date(5),date(6))
            if (self%StartTime%isValid()) then
            else
                self%StartTime = datetime() !reseting to default so it is caught later on
            end if
            sizem=sizeof(self%StartTime)
        else
            stop '[Globals::setparameter] StartTime parameter not in correct format. Eg. "2009 3 1 0 0 0"'
        end if
    elseif(parmkey%chars()=="EndTime") then
        call parmvalue%split(tokens=dc, sep=' ')
        if (size(dc) == 6) then
            do i=1, size(dc)
                date(i) = dc(i)%to_number(kind=1._r4p)
            end do
            self%EndTime = datetime(date(1),date(2),date(3),date(4),date(5),date(6))
            if (self%EndTime%isValid()) then
            else
                self%EndTime = datetime() !reseting to default so it is caught later on
            end if
            sizem=sizeof(self%EndTime)
        else
            stop '[Globals::setparameter] EndTime parameter not in correct format. Eg. "2009 3 1 0 0 0"'
        end if
    elseif(parmkey%chars()=="OutputFormat") then
        self%OutputFormat=parmvalue%to_number(kind=1_i1p)
        sizem=sizeof(self%OutputFormat)
    end if
    call simmemory%adddef(sizem)
```

6.13.2.19 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	*self,read_rho*	

Definition at line 472 of file simulation_globals.f90.

```
    implicit none
    class(constants_t), intent(inout) :: self
    type(string), intent(in) :: read_rho
    type(string) :: outext
    integer :: sizem
    self%Rho_ref=read_rho%to_number(kind=1._r4p)
    if (self%Rho_ref.le.0.0) then
        outext='Rho_ref must be positive and non-zero, stopping'
        call log%put(outext)
        stop
    endif
    sizem = sizeof(self%Rho_ref)
    call simmemory%adddef(sizem)
```

6.13.2.20 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	*self,read_z0*	

Definition at line 456 of file simulation_globals.f90.

```
456     implicit none
457     class(constants_t), intent(inout) :: self
458     type(string), intent(in) :: read_z0
459     integer :: sizem
460     self%Z0=read_z0%to_number(kind=1._r4p)
461     sizem = sizeof(self%Z0)
462     call simmemory%adddef(sizem)
```

6.13.3 Variable Documentation

6.13.3.1 globals

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

Definition at line 123 of file simulation_globals.f90.

```
123     type(globals_class) :: Globals
```

6.14 simulation_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.

6.14.1 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

6.14.2 Function/Subroutine Documentation

6.14.2.1 init_caseconstants()

```
subroutine simulation_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

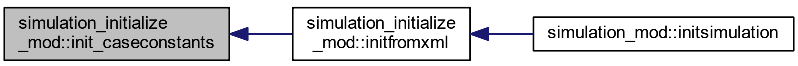
Parameters

in	*case_node*	

Definition at line 324 of file simulation_initialize_mod.f90.

```
324     implicit none
325     type(Node), intent(in), pointer :: case_node
326
327     type(Node), pointer :: constants_node
328     type(string) :: outext
329     type(string) :: tag, att_name, att_val
330     type(vector) :: coords
331     logical :: readflag
332
333     outext='-->Reading case constants'
334     call log%put(outext,.false.)
335
336     tag="constantsdef"    !the node we want
337     call xmlreader%gotoNode(case_node,constants_node,tag,readflag,.false.)
338     if (readflag) then !if the node exists, since his one is not mandatory
339       tag="Gravity"
340       call xmlreader%getNodeVector(constants_node,tag,coords,readflag,.false.)
341       if (readflag) then
342         call globals%Constants%setgravity(coords)
343       endif
344       tag="Z0"
345       att_name="value"
346       call xmlreader%getNodeAttribute(constants_node, tag, att_name, att_val,readflag,.false.)
347       if (readflag) then
348         call globals%Constants%setz0(att_val)
349       endif
350       tag="Rho_ref"
351       att_name="value"
352       call xmlreader%getNodeAttribute(constants_node, tag, att_name, att_val,readflag,.false.)
353       if (readflag) then
354         call globals%Constants%setrho(att_val)
355       endif
356     endif
357     call globals%Constants%print()
358
```

Here is the caller graph for this function:



6.14.2.2 init_parameters()

```
subroutine simulation_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

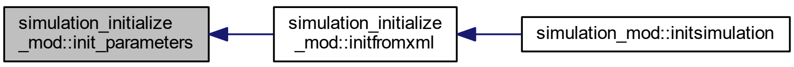
Parameters

in	*execution_node*	

Definition at line 368 of file simulation_initialize_mod.f90.

```
368     implicit none
369     type(Node), intent(in), pointer :: execution_node
370
371     type(string) :: outext
372     type(NodeList), pointer :: parameterList
373     type(Node), pointer :: parmt, parameters_node
374     integer :: i
375     type(string) :: parmkey, parmvalue, tag, att_name
376     character(80) :: parmkey_char, parmvalue_char
377
378     outext='-->Reading case parameters'
379     call log%put(outext,.false.)
380
381     tag="parameters"    !the node we want
382     call xmlreader%gotoNode(execution_node,parameters_node,tag)
383     parameterlist => getelementsbytagname(parameters_node, "parameter")       !searching for tags with the 'parameter' name
384     do i = 0, getlength(parameterlist) - 1                          !extracting parameter tags one by one
385         parmt => item(parameterlist, i)
386         att_name="key"
387         call xmlreader%getLeafAttribute(parmt,att_name,parmkey)
388         att_name="value"
389         call xmlreader%getLeafAttribute(parmt,att_name,parmvalue)
390         call globals%Parameters%setparameter(parmkey,parmvalue)
391     enddo
392     call globals%Parameters%check()
393     call globals%Parameters%print()
394
```

Here is the caller graph for this function:



6.14.2.3 init_properties()

```
subroutine simulation_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

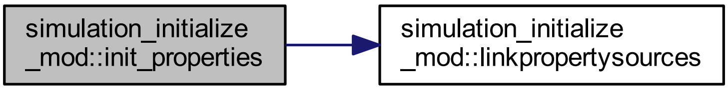
Parameters

in	*case_node*	

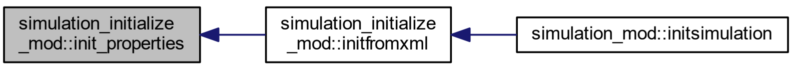
Definition at line 117 of file simulation_initialize_mod.f90.

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

    type(Node), pointer :: props_node
    type(string) :: outext
    type(string) :: tag, att_name

    tag="properties"    !the node we want
    call xmlreader%gotoNode(case_node,props_node,tag,mandatory =.false.)
    if (associated(props_node)) then
        tag="propertyfile"
        att_name="name"
        call xmlreader%getNodeAttribute(props_node, tag, att_name, globals%Names%propsxmlfilename) !getting the file name from that tag
        outext='-->Properties to link to Sources found at '//globals%Names%propsxmlfilename
        call log%put(outext,.false.)
        tag="links"
        call xmlreader%gotoNode(props_node,props_node,tag) !getting the links node
        call linkpropertysources(props_node)          !calling the property linker
    else
        outext='-->No properties to link to Sources, assuming pure Lagrangian tracers'
        call log%put(outext,.false.)
    endif
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.14.2.4 init_simdefs()

```
subroutine simulation_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

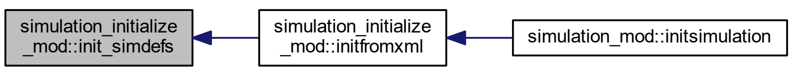
Parameters

in	*case_node*	

Definition at line 285 of file simulation_initialize_mod.f90.

```
285     implicit none
286     type(Node), intent(in), pointer :: case_node
287 
288     type(NodeList), pointer :: defsList
289     type(Node), pointer :: simdefs_node
290     type(string) :: outext
291     integer :: i
292     type(string) :: pts(2), tag, att_name, att_val
293     type(vector) :: coords
294 
295     outext='-->Reading case simulation definitions'
296     call log%put(outext,.false.)
297 
298     tag="simulationdefs"    !the node we want
299     call xmlreader%gotoNode(case_node,simdefs_node,tag)
300     tag="resolution"
301     att_name="dp"
302     call xmlreader%getNodeAttribute(simdefs_node, tag, att_name, att_val)
303     call globals%SimDefs%setdp(att_val)
304     tag="timestep"
305     att_name="dt"
306     call xmlreader%getNodeAttribute(simdefs_node, tag, att_name, att_val)
307     call globals%SimDefs%setdt(att_val)
308     pts=(/ 'pointmin', 'pointmax'/) !strings to search for
309     do i=1, size(pts)
310         call xmlreader%getNodeVector(simdefs_node, pts(i), coords)
311         call globals%SimDefs%setboundingbox(pts(i), coords)
312     enddo
313     call globals%SimDefs%print()
314 
```

Here is the caller graph for this function:



6.14.2.5 init_sources()

```
subroutine simulation_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

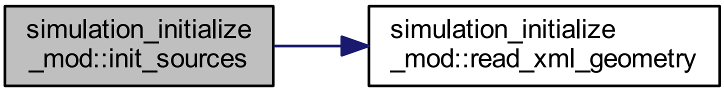
in	*case_node*	

Definition at line 192 of file simulation_initialize_mod.f90.

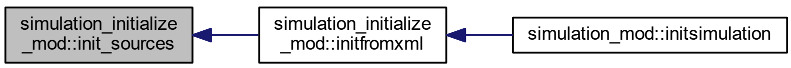
```
192     implicit none
193     type(Node), intent(in), pointer :: case_node
194 
195     type(string) :: outext
196     type(NodeList), pointer :: sourceList
197     type(NodeList), pointer :: sourceChildren
198     type(Node), pointer :: sourcedef
199     type(Node), pointer :: source_node
200     type(Node), pointer :: source_detail
201     integer :: i, j
202     logical :: readflag
203     !source vars
204     integer :: id
205     type(string) :: name, source_geometry, tag, att_name, att_val
206     real(prec) :: emitting_rate, start, finish
207     class(shape), allocatable :: source_shape
208 
209     outext='-->Reading case Sources'
210     call log%put(outext,.false.)
211 
212     tag="sourcedef"    !the node we want
213     call xmlreader%gotoNode(case_node,sourcedef,tag)
214     sourcelist => getelementsbytagname(sourcedef, "source")
215 
216     !allocating the temporary source objects
217     call tempsources%initialize(getlength(sourcelist))
218 
219     do j = 0, getlength(sourcelist) - 1
220         source_node => item(sourcelist,j)
221         tag="setsource"
222         att_name="id"
223         call xmlreader%getNodeAttribute(source_node, tag, att_name, att_val)
224         id=att_val%to_number(kind=1_i1p)
225         att_name="name"
226         call xmlreader%getNodeAttribute(source_node, tag, att_name, name)
227         tag="rate"
228         att_name="value"
229         call xmlreader%getNodeAttribute(source_node, tag, att_name, att_val)
230         emitting_rate = att_val%to_number(kind=1._r4p)
231         tag="active"
232         att_name="start"
233         call xmlreader%getNodeAttribute(source_node, tag, att_name, att_val,readflag,.false.)
234         if (readflag) then
235             start = att_val%to_number(kind=1._r4p)
236         else
237             start = 0.0
238         endif
239         att_name="end"
240         call xmlreader%getNodeAttribute(source_node, tag, att_name, att_val,readflag,.false.)
241         if (readflag.and.att_val%is_number()) then
242             finish = att_val%to_number(kind=1._r4p)
243         else
244             finish = globals%Parameters%TimeMax
245         endif
246         !now we need to find out the geometry of the source and read accordingly
247         sourcechildren => getchildnodes(source_node) !getting all of the nodes bellow the main source node
    (all of it's private info)
248         do i=0, getlength(sourcechildren)-1
249             source_detail => item(sourcechildren,i) !grabing a node
250             source_geometry = getlocalname(source_detail)  !finding its name
251             if (geometry%inlist(source_geometry)) then  !if the node is a valid geometry name
252                 select case (source_geometry%chars())
253                 case ('point')
254                     allocate(point::source_shape)
255                 case ('sphere')
256                     allocate(sphere::source_shape)
257                 case ('box')
258                     allocate(box::source_shape)
259                 case ('line')
260                     allocate(line::source_shape)
261                     case default
262                     outext='[init_sources]: unexpected type for geometry object!'
263                     call log%put(outext)
264                     stop
265                 end select
266                 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:



Here is the caller graph for this function:



6.14.2.6 initfromxml()

```
subroutine, public simulation_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

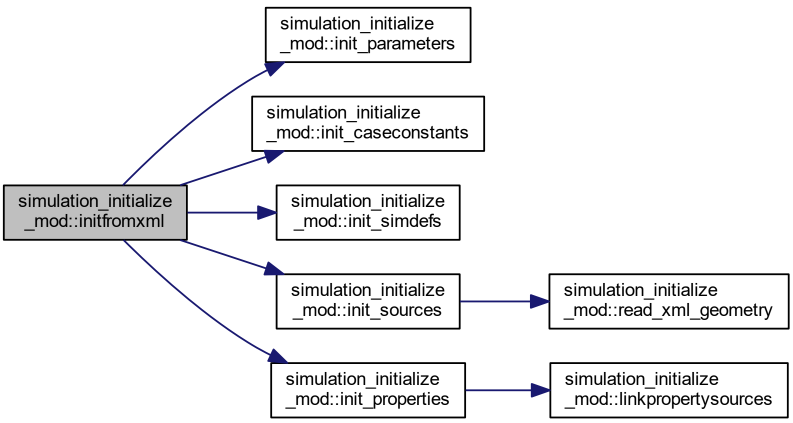
Parameters

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

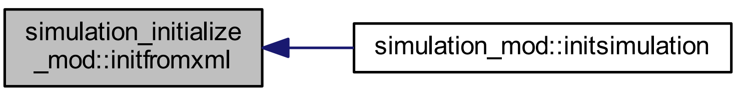
Definition at line 404 of file simulation_initialize_mod.f90.

```
    implicit none
    type(string), intent(in) :: xmlfilename
    type(string) :: outext, tag
    type(Node), pointer :: xmldoc
    type(Node), pointer :: case_node
    type(Node), pointer :: execution_node

    call xmlreader%getFile(xmldoc,xmlfilename)
    globals%Names%mainxmlfilename = xmlfilename
    globals%Names%casename = xmlfilename%basename(extension='.xml')
    outext='->Case name is '//globals%Names%casename
    call log%put(outext)

    tag="case"          !base document node
    call xmlreader%gotoNode(xmldoc,execution_node,tag)
    tag="execution"     !finding execution node
    call xmlreader%gotoNode(execution_node,execution_node,tag)
    tag="case"          !base document node
    call xmlreader%gotoNode(xmldoc,case_node,tag)
    tag="casedef"     !finding execution node
    call xmlreader%gotoNode(case_node,case_node,tag)

    ! building the simulation basic structures according to the case definition file
    ! every other structure in the simulation is built from these, i.e., not defined by the user directly
    call init_parameters(execution_node)
    call init_caseconstants(case_node)
    call init_simdefs(case_node)
    call init_sources(case_node)
    call init_properties(case_node)

    call xmlreader%closeFile(xmldoc)
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.14.2.7 linkpropertysources()

```
subroutine simulation_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

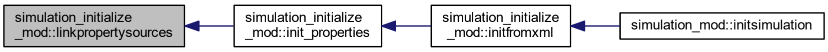
in	*linksNode*	

Definition at line 47 of file simulation_initialize_mod.f90.

```
47     implicit none
48     type(Node), intent(in), pointer :: linksNode
49
50     type(NodeList), pointer :: linkList
51     type(Node), pointer :: anode
52     type(Node), pointer :: xmlProps
53     type(string) :: xmlfilename, outext
54     integer :: i, p
55     type(string) :: att_name, att_val, tag
56     type(string) :: sourceid, sourcetype, sourceprop
57
58     linklist => getelementsbytagname(linksnode, "link")
59     do i = 0, getlength(linklist) - 1
60         anode => item(linklist,i)
61         att_name="source"
62         call xmlreader%getLeafAttribute(anode,att_name,sourceid)
63         att_name="type"
64         call xmlreader%getLeafAttribute(anode,att_name,sourcetype)
65         att_name="property"
66         call xmlreader%getLeafAttribute(anode,att_name,sourceprop)
67         !find the source and save the type and property name
68         call tempsources%setPropertyNames(sourceid,sourcetype,sourceprop)
69     enddo
70
71     !parse the properties file
72     xmlfilename = globals%Names%propsxmlfilename
73     call xmlreader%getFile(xmlprops,xmlfilename)
74
75     !Go to the materials node
76     tag = "materials"
77     call xmlreader%gotoNode(xmlprops,xmlprops,tag)
78
79     !find and set the actual atributes of the properties
80     att_name="value"
81     do i = 1, size(tempsources%src)
82         tag = tempsources%src(i)%prop%property_type
83         if (tag .ne. 'base') then
84         call xmlreader%gotoNode(xmlprops,anode,tag) !finding the material type node
85         tag = tempsources%src(i)%prop%property_name
86         call xmlreader%gotoNode(anode,anode,tag)      !finding the actual material node
87         do p = 1, size(globals%SrcProp%baselist)
88             call xmlreader%getNodeAttribute(anode, globals%SrcProp%baselist(p), att_name, att_val)

89             call tempsources%src(i)%setPropertyAtributes(globals%SrcProp%baselist(p), att_val)
90         end do
91         if (tempsources%src(i)%isParticulate()) then
92             do p = 1, size(globals%SrcProp%particulatelist)
93                 call xmlreader%getNodeAttribute(anode, globals%SrcProp%particulatelist(p), att_name,
   att_val)
94                 call tempsources%src(i)%setPropertyAtributes(globals%SrcProp%particulatelist(p), att_val)
95             end do
96         end if
97         !Run integrety check on the properties to see if Source is well defined
98         call tempsources%src(i)%check()
99         end if
```

```
    end do
    outext='-->Sources properties are set'
    call log%put(outext,.false.)

    call xmlreader%closeFile(xmlprops)

```

Here is the caller graph for this function:



6.14.2.8 read_xml_geometry()

```
subroutine simulation_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

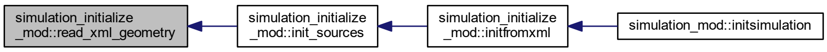
in	*source,source_detail,source_shape*	
in	*source*	Working xml node
in	*source_detail*	Working xml node details
in,out	*source_shape*	Geometrical object to fill

Definition at line 150 of file simulation_initialize_mod.f90.

```
150     implicit none
151     type(Node), intent(in), pointer :: source
152     type(Node), intent(in), pointer :: source_detail
153     class(shape), intent(inout) :: source_shape
154     type(string) :: outext
155     type(string) :: tag
156
157     select type (source_shape)
158     type is (shape)
159         !nothing to do
160     class is (box)
161         tag='point'
162         call xmlreader%getNodeVector(source_detail,tag,source_shape%pt)
163         tag='size'
164         call xmlreader%getNodeVector(source_detail,tag,source_shape%size)
165     class is (point)
166         tag='point'
167         call xmlreader%getNodeVector(source,tag,source_shape%pt)
168     class is (line)
```

```
        tag='pointa'
        call xmlreader%getNodeVector(source_detail,tag,source_shape%pt)
        tag='pointb'
        call xmlreader%getNodeVector(source_detail,tag,source_shape%last)
    class is (sphere)
        tag='point'
        call xmlreader%getNodeVector(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 caller graph for this function:



6.15 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

6.15.1 Detailed Description

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

Author

Ricardo Birjukovs Canelas

6.15.2 Function/Subroutine Documentation

6.15.2.1 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 72 of file simulation_logger.f90.

```
72     implicit none
73     class(logger_class), intent(inout) :: self
74     close(self%log_unit)
```

6.15.2.2 gettimestamp()

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

Public timestamp builder.

Author

Ricardo Birjukovs Canelas - MARETEC

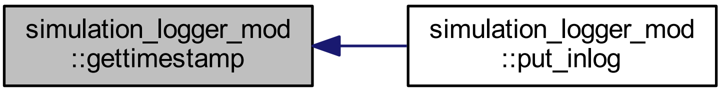
Parameters

in	*timestamp*	

Definition at line 109 of file simulation_logger.f90.

```
109     implicit none
110     type(string), intent(out) :: timestamp
111     character(80) :: temp(8)
112     integer :: values(8),i
113
114     call date_and_time(values=values)
115     do i=1,8
116         write(temp(i),*) values(i)
117     enddo
118     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:



6.15.2.3 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	*self,outpath*	
in	*outpath*	output path were to point the logger

Definition at line 55 of file simulation_logger.f90.

```
55     implicit none
56     class(logger_class), intent(inout) :: self
57     type(string), intent(in) :: outpath
58     type(string) :: logfile
59
60     logfile = outpath//'MOHIDLagrangianRun.out'
61     self%log_unit = 0
62     open (unit=self%log_unit,file=logfile%chars(),action="write",status="replace")
63
```

6.15.2.4 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

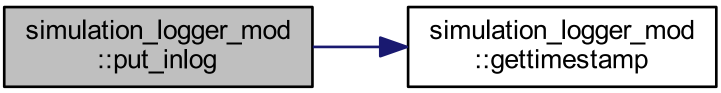
Parameters

in	*self,tologstr,timeoption*	

Definition at line 84 of file simulation_logger.f90.

```
84      implicit none
85      class(logger_class), intent(in) :: self
86      type(string), intent(inout) :: tologstr
87      logical, intent(in), optional :: timeoption
88      type(string) :: timestamp
89
90      call gettimestamp(timestamp)
91      if (present(timeoption)) then
92        if (.not.timeoption) then
93          timestamp=''
94        endif
95      endif
96      tologstr=timestamp//' '//tologstr
97      write(self%log_unit,"(A)") tologstr%chars()
98      print'(A)', tologstr%chars()
99
```

Here is the call graph for this function:



6.15.3 Variable Documentation

6.15.3.1 log

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

Definition at line 38 of file simulation_logger.f90.

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

6.16 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)

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

  Method to retreive the total size of the allocated memory.
- subroutine setntrc (self, ntrc)

  Method to set the total expected number of Tracers.
- subroutine setsizetrc (self, sizeTrc)

  Method to set the size of a typical Tracer.
- subroutine addblock (self, size)

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

  Method to add the size of a Source to the memory log.
- subroutine setracer (self, size)

  Method to add the size of a Tracer to the memory log.
- subroutine adddef (self, size)

  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)

  Method to print the allocated memory.

Variables

- type(memory_t), public simmemory

6.16.1 Detailed Description

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

Author

Ricardo Birjukovs Canelas

6.16.2 Function/Subroutine Documentation

6.16.2.1 addblock()

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

Method to add the size of a Block to the memory log.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 113 of file simulation_memory.f90.

```
113     implicit none
114     class(memory_t), intent(inout) :: self
115     integer, intent(in) :: size
116     self%size_of_blocks = self%size_of_blocks + size
```

6.16.2.2 adddef()

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

Method to add the size of a definition to the memory log.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 150 of file simulation_memory.f90.

```
150     implicit none
151     class(memory_t), intent(inout) :: self
152     integer, intent(in) :: size
153     self%size_of_defs = self%size_of_defs + size
```

6.16.2.3 addsource()

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

Method to add the size of a Source to the memory log.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 125 of file simulation_memory.f90.

```
125     implicit none
126     class(memory_t), intent(inout) :: self
127     integer, intent(in) :: size
128     self%size_of_sources = self%size_of_sources + size
```

6.16.2.4 getotal()

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

6.16.2.5 initializememory()

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

Memory logger initialization method.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 63 of file simulation_memory.f90.

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

6.16.2.6 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 162 of file simulation_memory.f90.

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

6.16.2.7 printmemorydetailed()

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

Method to print the allocated memory.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 180 of file simulation_memory.f90.

```
180     implicit none
181     class(memory_t), intent(inout) :: self
182     integer :: size
183     real(prec) :: sizemb
184     type(string) :: outext,temp(6)
185     call self%getotal(size)
186     sizemb = size*1e-6
187     temp(1)= sizemb
188     sizemb = self%size_of_sources*1e-6
189     temp(2)= sizemb
190     sizemb = self%size_of_tracers*1e-6
191     temp(3)= sizemb
192     sizemb = self%size_of_defs*1e-6
193     temp(4)= sizemb
194     sizemb = self%size_of_blocks*1e-6
195     temp(5)= sizemb
196     sizemb = self%ntrc*self%sizeTrc*1e-6
197     temp(6) = 2.25*sizemb
198     outext='->Total allocated memory: '//temp(1)//' mb'//new_line('a')//&
199         '       Allocated memory for Blocks   = '//temp(5)//' mb'//new_line('a')//&
200         '       Allocated memory for Sources = '//temp(2)//' mb'//new_line('a')//&
201         '       Allocated memory for Tracers = '//temp(3)//' mb'//new_line('a')//&
202         '       Allocated memory for Consts  = '//temp(4)//' mb'//new_line('a')//&
203         '       Expected memory requirements exceed '//temp(6)//' mb'
204     call log%put(outext)
```

6.16.2.8 setntrc()

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

Method to set the total expected number of Tracers.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 89 of file simulation_memory.f90.

```
89     implicit none
90     class(memory_t), intent(inout) :: self
91     integer, intent(in) :: ntrc
92     self%ntrc = ntrc
```

6.16.2.9 setracer()

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

Method to add the size of a Tracer to the memory log.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 137 of file simulation_memory.f90.

```
137     implicit none
138     class(memory_t), intent(inout) :: self
139     integer, intent(in) :: size
140     self%size_of_tracers = size
```

6.16.2.10 setsizetrc()

```
subroutine simulation_memory_mod::setsizetrc (
            class(memory_t), intent(inout) self,
            integer*8, intent(in) sizeTrc ) [private]
```

Method to set the size of a typical Tracer.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 101 of file simulation_memory.f90.

```
101     implicit none
102     class(memory_t), intent(inout) :: self
103     integer*8, intent(in) :: sizeTrc
104     self%sizeTrc = sizetrc
```

6.16.3 Variable Documentation

6.16.3.1 simmemory

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

Definition at line 50 of file simulation_memory.f90.

```
50     type(memory_t) :: SimMemory
```

6.17 simulation_mod Module Reference

Module to hold the simulation class and its methods. This is the only class that is exposed to an external program, as it encapsulates every other class and method.

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 togglesources (self)

  Simulation method to activate and deactivate Sources based on the GlobalSimTime.
- subroutine blocksemitt (self)

  Simulation method to call the Blocks to emitt tracers at current SimTime.
- subroutine blocksdistribute (self)

  Simulation method to call the Blocks to distribute Tracers at current SimTime.
- subroutine blocksconsolidatearrays (self)

  Simulation method to call the Blocks to consolidate the Tracer array at current SimTime.
- subroutine blockstracerstoaot (self)

  Simulation method to call the Blocks to build their Array of Tracers (AoT) from the Tracer list at current SimTime.
- subroutine blocksaottotracers (self)

  Simulation method to call the Blocks to print their Array of Tracers (AoT) back to the Tracer objects on the list at current SimTime.
- subroutine blockscleanaot (self)

  Simulation method to call the Blocks to clean their Array of Tracers (AoT) at current SimTime.
- subroutine setinitialstate (self)

  Simulation method to distribute the Sources to the Blocks, allocate the respective Tracers and redistribute if needed.
- integer function gettracertotals (self)

  Simulation method to count Tracer numbers.
- subroutine printtracertotals (self)

  Simulation method to count Tracer numbers.
- subroutine settracermemory (self, ntrc)

  Simulation method to account for Tracer memory consumption.
- 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.

6.17.1 Detailed Description

Module to hold the simulation class and its methods. This is the only class that is exposed to an external program, as it encapsulates every other class and method.

Author

Ricardo Birjukovs Canelas

6.17.2 Function/Subroutine Documentation

6.17.2.1 blocksaottotracers()

```
subroutine simulation_mod::blocksaottotracers (
            class(simulation_class), intent(in) self )  [private]
```

Simulation method to call the Blocks to print their Array of Tracers (AoT) back to the Tracer objects on the list at current SimTime.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 251 of file simulation.f90.

```
251         implicit none
252         class(simulation_class), intent(in) :: self
253         integer :: i
254         do i=1, size(dblock)
255             call dblock(i)%AoTtoTracers()
256         enddo
```

6.17.2.2 blockscleanaot()

```
subroutine simulation_mod::blockscleanaot (
            class(simulation_class), intent(in) self )  [private]
```

Simulation method to call the Blocks to clean their Array of Tracers (AoT) at current SimTime.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 266 of file simulation.f90.

```
266         implicit none
267         class(simulation_class), intent(in) :: self
268         integer :: i
269         do i=1, size(dblock)
270             call dblock(i)%CleanAoT()
271         enddo
```

6.17.2.3 blocksconsolidatearrays()

subroutine simulation_mod::blocksconsolidatearrays (
class(simulation_class), intent(in) *self*) [private]

Simulation method to call the Blocks to consolidate the Tracer array at current SimTime.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 221 of file simulation.f90.

```
221         implicit none
222         class(simulation_class), intent(in) :: self
223         integer :: i
224         do i=1, size(dblock)
225             call dblock(i)%ConsolidateArrays()
226         enddo
```

6.17.2.4 blocksdistribute()

subroutine simulation_mod::blocksdistribute (
class(simulation_class), intent(in) *self*) [private]

Simulation method to call the Blocks to distribute Tracers at current SimTime.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 205 of file simulation.f90.

```
205         implicit none
206         class(simulation_class), intent(in) :: self
207         integer :: i
208         do i=1, size(dblock)
209             call dblock(i)%DistributeTracers()
210         enddo
211         !need to distribute Sources also! TODO
```

6.17.2.5 blocksemitt()

subroutine simulation_mod::blocksemitt (
class(simulation_class), intent(in) *self*) [private]

Simulation method to call the Blocks to emitt tracers at current SimTime.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 190 of file simulation.f90.

```
190         implicit none
191         class(simulation_class), intent(in) :: self
192         integer :: i
193         do i=1, size(dblock)
194             call dblock(i)%CallEmitter()
195         enddo
```

6.17.2.6 blockstracerstoaot()

```
subroutine simulation_mod::blockstracerstoaot (
            class(simulation_class), intent(in) self )  [private]
```

Simulation method to call the Blocks to build their Array of Tracers (AoT) from the Tracer list at current SimTime.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 236 of file simulation.f90.

```
236         implicit none
237         class(simulation_class), intent(in) :: self
238         integer :: i
239         do i=1, size(dblock)
240             call dblock(i)%TracersToAoT()
241         enddo
```

6.17.2.7 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 379 of file simulation.f90.

```
379     implicit none
380     class(simulation_class), intent(inout) :: self
381     type(string) :: outext
382     outext='Simulation ended, freeing resources. See you next time'
383     call log%put(outext)
384     call log%finalize()
```

6.17.2.8 decomposedomain()

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

Simulation method to do domain decomposition and define the Blocks.

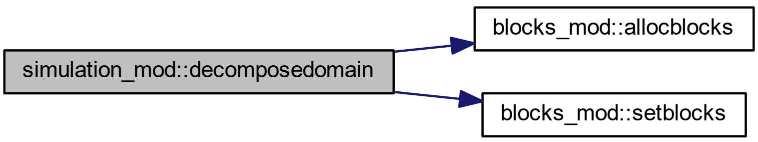
Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 359 of file simulation.f90.

```
359     implicit none
360     class(simulation_class), intent(inout) :: self
361     type(string) :: outext
362     if (globals%SimDefs%autoblocksize) then
363         call allocblocks(globals%SimDefs%numblocks)
364     else
365         outext='[DecomposeDomain]: Only automatic Block sizing at the moment, stoping'
366         call log%put(outext)
367         stop
368     end if
369     ! Initializing the Blocks
370     call setblocks(globals%SimDefs%autoblocksize,globals%SimDefs%numblocks,globals%SimDefs%numblocksx,
      globals%SimDefs%numblocksy)
```

Here is the call graph for this function:



6.17.2.9 gettracertotals()

integer function simulation_mod::gettracertotals (
    class(simulation_class), intent(in) *self*) [private]

Simulation method to count Tracer numbers.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 307 of file simulation.f90.

```
307     implicit none
308     class(simulation_class), intent(in) :: self
309     integer :: i, total
310     total = 0
311     do i=1, size(dblock)
312         total = total + dblock(i)%numAllocTracers()
313     enddo
314     gettracertotals = total
```

6.17.2.10 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

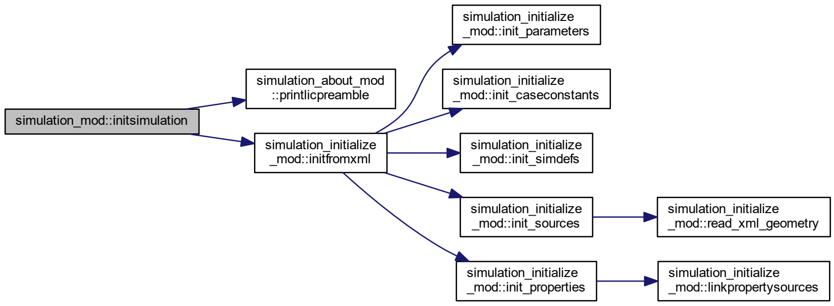
Parameters

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

Definition at line 123 of file simulation.f90.

```
123     implicit none
124     class(simulation_class), intent(inout) :: self
125     type(string), intent(in) :: casefilename
126     type(string), intent(in) :: outpath
127     type(string) :: outext
128     !type(generic_field_class) :: testField
129     !type(background_class) :: testBackground
130
131     ! Initialize logger
132     call log%initialize(outpath)
133     !Print licences and build info
134     call printlicpreamble
135     !initializing memory log
136     call simmemory%initialize()
137     !setting every global variable and input parameter to their default
138     call globals%initialize(outpath = outpath)
139     !initializing geometry class
140     call geometry%initialize()
141     !Check if case file has .xml extension
142     if (casefilename%extension() == '.xml') then
143         ! Initialization routines to build the simulation from the input case file
144         call initfromxml(casefilename)
145     else
146         outext='[initSimulation]: only .xml input files are supported at the time. Stopping'
147         call log%put(outext)
148         stop
149     endif
150     !Case was read and now we can build/initialize our simulation objects that are case-dependent
151     !initilize simulation bounding box
152     call bbox%initialize()
153     !decomposing the domain and initializing the Simulation Blocks
154     call self%decompose()
155     !Distributing Sources
156     call self%setInitialState()
157     !printing memory occupation at the time
158     call simmemory%detailedprint()
159     !Initializing output file streamer
160     call outputstreamer%initialize()
161     !Writing the domain to file
162     call outputstreamer%WriteDomain(globals%Names%casename, bbox, geometry%getnumPoints(bbox), dblock)
163
164     !call testField%test()
165     !call testBackground%test()
166
```

Here is the call graph for this function:



6.17.2.11 printtracertotals()

```
subroutine simulation_mod::printtracertotals (
            class(simulation_class), intent(in) self )  [private]
```

Simulation method to count Tracer numbers.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 323 of file simulation.f90.

```
323     implicit none
324     class(simulation_class), intent(in) :: self
325     type(string) :: outext, temp
326     temp = self%getTracerTotals()
327     outext='-->'//temp //' Tracers allocated'
328     call log%put(outext,.false.)
```

6.17.2.12 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 68 of file simulation.f90.

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

    outext = '=========================================================================='
    call log%put(outext,.false.)
    outext = '->Simulation staring'
    call log%put(outext)
    outext = '=========================================================================='
    call log%put(outext,.false.)

    !main time cycle
    do while (globals%SimTime .lt. globals%Parameters%TimeMax)
        !activate suitable Sources
        call self%ToggleSources()
        !emitt Tracers from active Sources
        call self%BlocksEmitt()
        !Distribute Tracers and Sources by Blocks
        call self%BlocksDistribute()
        !Optimize Block Tracer lists
        call self%BlocksConsolidateArrays()
        !Build AoT
        call self%BlocksTracersToAoT()
        !load hydrodynamic fields from files (curents, wind, waves, ...)
        !interpolate fields to tracer coordinates
        !Update all tracers with base behavior (AoT)
        !AoT to Tracers
        call self%BlocksAoTtoTracers()
        !Update Tracers with type-specific behavior
        !Write results if time to do so
        call outputstreamer%WriteStepSerial(dblock)
        !Print some stats from the time step
        call self%printTracerTotals()
        !Clean AoT
        call self%BlocksCleanAoT()
        !update Simulation time and counters
        globals%SimTime = globals%SimTime + globals%SimDefs%dt
        call globals%Sim%increment_numdt()
        !print*, 'Global time is ', Globals%SimTime
        !print*, 'Can we continue?'
        !read (*,*)
    enddo
    call self%setTracerMemory()
    call simmemory%detailedprint()

```

6.17.2.13 setinitialstate()

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

Simulation method to distribute the Sources to the Blocks, allocate the respective Tracers and redistribute if needed.

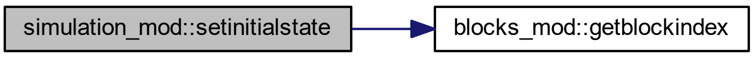
Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 281 of file simulation.f90.

```
    implicit none
    class(simulation_class), intent(inout) :: self
    type(string) :: outext
    integer :: i, blk, ntrc
    !iterate every Source to distribute
    ntrc = 0
    do i=1, size(tempsources%src)
        blk = getblockindex(geometry%getCenter(tempsources%src(i)%par%geometry))
        call dblock(blk)%putSource(tempsources%src(i))
        ntrc = ntrc + tempsources%src(i)%stencil%total_np
    end do
    call tempsources%finalize() !destroying the temporary Sources now they are shipped to the Blocks
    outext='-->Sources allocated to their current Blocks'
    call log%put(outext,.false.)
    outext = ntrc
    outext='-->'//outext//' Tracers on the emission stack'
    call log%put(outext,.false.)
    call self%setTracerMemory(ntrc)
```

Here is the call graph for this function:



6.17.2.14 settracermemory()

```
subroutine simulation_mod::settracermemory (
            class(simulation_class), intent(in) self,
            integer, intent(in), optional ntrc )  [private]
```

Simulation method to account for Tracer memory consumption.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 337 of file simulation.f90.

```
337    implicit none
338    class(simulation_class), intent(in) :: self
339    integer, optional, intent(in) :: ntrc
340    integer :: sizem, i
341    sizem = 0
342    do i=1, size(dblock)
343        sizem = sizem + sizeof(dblock(i)%LTracer) !this accounts for the array structure
344        sizem = sizem + sizeof(dummytracer)*dblock(i)%LTracer%getSize() !this accounts for the contents
345    enddo
346    call simmemory%setracer(sizem)
347    if(present(ntrc)) then
348        call simmemory%setNtrc(ntrc)
349        call simmemory%setsizeTrc(sizeof(dummytracer))
350    end if
```

6.17.2.15 togglesources()

```
subroutine simulation_mod::togglesources (
            class(simulation_class), intent(in) self )  [private]
```

Simulation method to activate and deactivate Sources based on the GlobalSimTime.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 176 of file simulation.f90.

```
176         implicit none
177         class(simulation_class), intent(in) :: self
178         integer :: i
179         do i=1, size(dblock)
180            call dblock(i)%ToogleBlockSources()
181         enddo
```

6.18 simulation_output_streamer_mod Module Reference

Defines a output file writer class with an object exposable to the Simulation This class is in charge of selectig the correct writter for the selected output file format.

Data Types

- type output_streamer_class

Functions/Subroutines

- subroutine initoutputstreamer (self)

    Initializes the Output writer object.
- subroutine writestepserial (self, blocks)

    Streamer method to call a simulation step writer. Writes binary XML VTK format using an unstructured grid.
- subroutine writedomain (self, filename, bbox, npbbox, blocks)

    Public simulation domain writting routine. Writes binary XML VTK format using an unstructured grid.

Variables

- type(output_streamer_class), public outputstreamer

6.18.1 Detailed Description

Defines a output file writer class with an object exposable to the Simulation This class is in charge of selectig the correct writter for the selected output file format.

Author

Ricardo Birjukovs Canelas

6.18.2 Function/Subroutine Documentation

6.18.2.1 initoutputstreamer()

```
subroutine simulation_output_streamer_mod::initoutputstreamer (
            class(output_streamer_class), intent(inout) self )  [private]
```

Initializes the Output writer object.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 54 of file simulation_output_streamer.f90.

```
54     implicit none
55     class(output_streamer_class), intent(inout) :: self
56     self%OutputFormat = globals%Parameters%OutputFormat
57     if (self%OutputFormat == 2) then !VTK file selected
58         call vtkwritter%initialize()
59     end if
```

6.18.2.2 writedomain()

```
subroutine simulation_output_streamer_mod::writedomain (
            class(output_streamer_class), intent(inout) self,
            type(string), intent(in) filename,
            class(boundingbox_class), intent(in) bbox,
            integer, intent(in) npbbox,
            class(block_class), dimension(:), intent(in) blocks )  [private]
```

Public simulation domain writting routine. Writes binary XML VTK format using an unstructured grid.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,filename,bbox,npbbox,blocks*	
in	*filename*	name of the case to add
in	*bbox*	Case bounding box
in	*npbbox*	number of points of the bbox geometry
in	*blocks*	Case Blocks

Definition at line 93 of file simulation_output_streamer.f90.

```
implicit none
class(output_streamer_class), intent(inout) :: self
type(string), intent(in) :: filename
class(boundingbox_class), intent(in) :: bbox
integer, intent(in) :: npbbox
class(block_class), dimension(:), intent(in) :: blocks

 if (self%OutputFormat == 2) then !VTK file selected
     call vtkwritter%Domain(filename, bbox, npbbox, blocks)
 end if

```

6.18.2.3 writestepserial()

```
subroutine simulation_output_streamer_mod::writestepserial (
            class(output_streamer_class), intent(inout) self,
            class(block_class), dimension(:), intent(in) blocks )  [private]
```

Streamer method to call a simulation step writer. Writes binary XML VTK format using an unstructured grid.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,blocks*	
in	*blocks*	Case Blocks

Definition at line 70 of file simulation_output_streamer.f90.

```
implicit none
class(output_streamer_class), intent(inout) :: self
class(block_class), dimension(:), intent(in) :: blocks
type(string) :: filename

filename = globals%Names%casename//'_'//int2str('(i5.5)',globals%Sim%getnumoutfile())

if (self%OutputFormat == 2) then !VTK file selected
    call vtkwritter%TracerSerial(filename, blocks)
end if

call globals%Sim%increment_numoutfile()

```

6.18.3 Variable Documentation

6.18.3.1 outputstreamer

```
type(output_streamer_class), public simulation_output_streamer_mod::outputstreamer
```

Definition at line 39 of file simulation_output_streamer.f90.

```
type(output_streamer_class) :: OutputStreamer
```

6.19 simulation_precision_mod Module Reference

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

Variables

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

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

  Double precision definition switch.
- integer, parameter, public prec = dps
- integer, parameter, public prec_time = sps
- integer, parameter, public prec_wrt = sps
- real(prec), parameter, public missing_value_default = -9999.0_dps
- 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_dps
- integer, parameter, public err_ind = -1
- integer, parameter, public char_len = 99

6.19.1 Detailed Description

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

Author

Ricardo Birjukovs Canelas

6.19.2 Variable Documentation

6.19.2.1 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
```

6.19.2.2 dps

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

Double precision definition switch.

Definition at line 31 of file simulation_precision.f90.

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

6.19.2.3 err_dist

```
real(prec), parameter, public simulation_precision_mod::err_dist = 1E8_dps
```

Definition at line 44 of file simulation_precision.f90.

```
44     real(prec), parameter :: ERR_DIST = 1e8_dps
```

6.19.2.4 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
```

6.19.2.5 missing_value_default

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

Definition at line 39 of file simulation_precision.f90.

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

6.19.2.6 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
```

6.19.2.7 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)
```

6.19.2.8 prec

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

Definition at line 34 of file simulation_precision.f90.

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

6.19.2.9 prec_time

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

Definition at line 35 of file simulation_precision.f90.

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

6.19.2.10 prec_wrt

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

Definition at line 36 of file simulation_precision.f90.

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

6.19.2.11 sps

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

Simple precision definition switch.

Definition at line 30 of file simulation_precision.f90.

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

6.20 sources_list_mod Module Reference

Module to hold the Sources linked list class and its methods. This class defines a double linked list to store any variable type, but with specific methods with type guards for Source objects. The class allows for insertion, deletion and iteration of the desired contents.

Data Types

- type sourcelist_class

Functions/Subroutines

- subroutine print_sourcelist (this)

    Method that prints all the links of the list.
- subroutine print_sourcelistcurrent (this)

    Method that prints the current link of the list.

6.20.1 Detailed Description

Module to hold the Sources linked list class and its methods. This class defines a double linked list to store any variable type, but with specific methods with type guards for Source objects. The class allows for insertion, deletion and iteration of the desired contents.

Author

Ricardo Birjukovs Canelas

6.20.2 Function/Subroutine Documentation

6.20.2.1 print_sourcelist()

```
subroutine sources_list_mod::print_sourcelist (
            class(sourcelist_class), intent(in) this )  [private]
```

Method that prints all the links of the list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 47 of file sources_list.f90.

```
    class(sourceList_class), intent(in) :: this
    class(*), pointer :: curr
    call this%reset()               ! reset list iterator
    do while(this%moreValues())     ! loop while there are values to print
        call this%printCurrent()
        call this%next()            ! increment the list iterator
    end do
    call this%reset()               ! reset list iterator
```

6.20.2.2 print_sourcelistcurrent()

```
subroutine sources_list_mod::print_sourcelistcurrent (
            class(sourcelist_class), intent(in) this )  [private]
```

Method that prints the current link of the list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 63 of file sources_list.f90.

```
    class(sourceList_class), intent(in) :: this
    class(*), pointer :: curr
    type(string) :: outext
    curr => this%currentValue() ! get current value
    select type(curr)
    class is (source_class)
        call curr%print()
        class default
        outext = '[sourceList_class::print] Unexepected type of content, not a Source'
        call log%put(outext)
        stop
    end select
```

6.21 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_prop

  Type - material properties of a source object.
- 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 linkproperty (src, ptype, pname)

  source property setting proceadure - initializes Source variables
- subroutine setpropertynames (self, srcid_str, ptype, pname)

  source property setting routine, calls source by id to set its properties
- subroutine setpropertyatributes (src, pname, pvalue)

  source property atribute setting proceadure - initializes Source variables
- subroutine check (self)

  Method that checks for the consistency of the Source properties.
- subroutine initializesource (src, id, name, emitting_rate, start, finish, source_geometry, shapetype)

  source inititialization proceadure - initializes Source variables
- logical function isparticulate (self)

  Returns particulate status of this Source, i.e, true if the emitted Tracers are actually a collection of particles with an evolving concentration.
- subroutine setotalnp (self)

  method that sets the total number of tracers a source will potentially create
- subroutine printsource (src)

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

Variables

- type(source_group_class), public tempsources

  Temporary Source array, used exclusively for building the case from a description file.

6.21.1 Detailed Description

Module that defines a source class and related methods.

Author

Ricardo Birjukovs Canelas

6.21.2 Function/Subroutine Documentation

6.21.2.1 check()

```
subroutine sources_mod::check (
            class(source_class), intent(in) self )  [private]
```

Method that checks for the consistency of the Source properties.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 241 of file sources.f90.

```
241     implicit none
242     class(source_class), intent(in) :: self
243     type(string) :: outext, temp(2)
244     logical :: failed
245     failed = .false.
246     temp(1) = self%par%id
247     if (self%prop%radius == mv) then
248         failed = .true.
249         temp(2) = 'radius'
250     elseif (self%prop%density == mv) then
251         failed = .true.
252         temp(2) = 'density'
253     elseif (self%prop%condition == mv) then
254         failed = .true.
255         temp(2) = 'condition'
256     elseif (self%prop%degrd_rate == mv) then
257         failed = .true.
258         temp(2) = 'degradation rate'
259     elseif (self%prop%particulate) then
260         if (self%prop%pt_radius == mv) then
261             failed = .true.
262             temp(2) = 'particle radius'
263         elseif (self%prop%ini_concentration == mv) then
264             failed = .true.
265             temp(2) = 'initial concentration'
266         end if
267     end if
268     if (failed) then
269         outext = 'Source'//temp(1)//' '//temp(2)//' is not set, stoping'
270         call log%put(outext)
271         stop
272     end if
```

6.21.2.2 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,start,finish,source_geometry,shapetype*	

Definition at line 282 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)
    !Setting properties
    src%prop%property_type = "base" ! pure Lagrangian trackers by default
    src%prop%property_name = "base"
    src%prop%particulate = .false.
    src%prop%radius = mv
    src%prop%density = mv
    src%prop%condition = mv
    src%prop%degrd_rate = mv
    src%prop%pt_radius = mv
    src%prop%ini_concentration = mv
    !Setting state variables
    src%now%age=0.0
    src%now%active=.false. !disabled by default
    src%now%emission_stride=1 !first time-step once active the Source emitts
    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, globals%SimDefs%Dp)
    call src%setotalnp()
    allocate(src%stencil%ptlist(src%stencil%np), stat=err)
    if(err/=0)then
        outext='[Sources::initialize]:Cannot allocate point list for Source '// src%par%name //', stoping'
        call log%put(outext)
        stop
    endif
    call geometry%fill(src%par%geometry, globals%SimDefs%Dp, src%stencil%np, src%stencil%ptlist)
    do i=1, src%stencil%np
            src%stencil%ptlist(i) = m2geo(src%stencil%ptlist(i), src%stencil%ptlist(i)%y)
    end do


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

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

6.21.2.3 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	*self,nsources*	

Definition at line 112 of file sources.f90.

```
112     implicit none
113     class(source_group_class), intent(inout) :: self
114     integer, intent(in) :: nsources
115     integer err
116     type(string) :: outext, temp
117     allocate(self%src(nsources), stat=err)
118     if(err/=0)then
119         outext='[initSources]: Cannot allocate Sources, stoping'
120         call log%put(outext)
121         stop
122     else
123         temp = nsources
124         outext = 'Allocated '// temp // ' Sources.'
125         call log%put(outext)
126     endif
```

6.21.2.4 isparticulate()

```
logical function sources_mod::isparticulate (
            class(source_class) self )  [private]
```

Returns particulate status of this Source, i.e, true if the emitted Tracers are actually a collection of particles with an evolving concentration.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 356 of file sources.f90.

```
356     class(source_class) :: self
357     isparticulate = self%prop%particulate
```

6.21.2.5 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 136 of file sources.f90.

```
136     implicit none
137     class(source_group_class), intent(inout) :: self
138     integer err
139     type(string) :: outext
140     if (allocated(self%src)) deallocate(self%src, stat=err)
141     if(err/=0)then
142         outext='[killSources]: Cannot deallocate Sources, stoping'
143         call log%put(outext)
144         stop
145     endif
```

6.21.2.6 linkproperty()

```
subroutine sources_mod::linkproperty (
            class(source_class), intent(inout) 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 155 of file sources.f90.

```
155     implicit none
156     class(source_class), intent(inout) :: src
157     type(string), intent(in) :: ptype
158     type(string), intent(in) :: pname
159     src%prop%property_type = ptype
160     src%prop%property_name = pname
```

6.21.2.7 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 379 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//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
    temp_str(3)=src%stencil%total_np
    outext = outext//'       Emitting '//temp_str(2)//' tracers at every '//temp_str(1)//' time-steps'//
  new_line('a')
    outext = outext//'       For an estimated total of '//temp_str(3)//' tracers' //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.)

```

6.21.2.8 setotalnp()

```
subroutine sources_mod::setotalnp (
            class(source_class), intent(inout) self )  [private]
```

method that sets the total number of tracers a source will potentially create

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in,out	*self*	computing the total as $NP^{source-i}_{total} = int((T^{source-i}_{end} - T^{source-i}_{start})/(Dt/Rate^{source-i}) * NP^{source-i}_{emission})$

Definition at line 366 of file sources.f90.

```
366     implicit none
367     class(source_class), intent(inout) :: self
368
369     self%stencil%total_np=int((self%par%stoptime-self%par%startime)/(globals%SimDefs%dt)/self%par
     %emitting_rate*self%stencil%np)
```

6.21.2.9 setpropertyatributes()

```
subroutine sources_mod::setpropertyatributes (
            class(source_class), intent(inout) src,
            type(string), intent(in) pname,
            type(string), intent(in) pvalue )  [private]
```

source property atribute setting proceadure - initializes Source variables

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*src,pname,pvalue*	

Definition at line 207 of file sources.f90.

```
207     implicit none
208     class(source_class), intent(inout) :: src
209     type(string), intent(in) :: pname
210     type(string), intent(in) :: pvalue
211     type(string) :: outext
212     select case (pname%chars())
213     case ('particulate')
214         if (pvalue%to_number(kind=1_i1p) == 1) then
215             src%prop%particulate = .true.
216         end if
217     case ('radius')
218         src%prop%radius = pvalue%to_number(kind=1._r4p)
219     case ('particle_radius')
220         src%prop%pt_radius = pvalue%to_number(kind=1._r4p)
221     case ('density')
222         src%prop%density = pvalue%to_number(kind=1._r4p)
223     case ('condition')
224         src%prop%condition = pvalue%to_number(kind=1._r4p)
225     case ('degradation_rate')
226         src%prop%degrd_rate = pvalue%to_number(kind=1._r4p)
227     case ('initial_concentration')
228         src%prop%ini_concentration = pvalue%to_number(kind=1._r4p)
229         case default
230         outext='[Sources::setPropertyAtributes]: unexpected atribute '//pname//' for property '//src%prop
     %property_name//', ignoring'
231         call log%put(outext)
232     end select
```

6.21.2.10 setpropertynames()

```
subroutine sources_mod::setpropertynames (
            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	*self,srcid_str,ptype,pname*	
in	*srcid_str*	Source id tag
in	*ptype*	Property type to set
in	*pname*	Property name to set

Definition at line 170 of file sources.f90.

```
170     implicit none
171     class(source_group_class), intent(inout) :: self
172     type(string), intent(in) :: srcid_str
173     type(string), intent(in) :: ptype
174     type(string), intent(in) :: pname
175
176     integer :: srcid
177     type(string) :: outext, temp
178     integer :: i
179     logical :: notlinked
180
181     srcid = srcid_str%to_number(kind=1_i1p)
182     notlinked = .true.  !assuming not linked
183     do i=1, size(self%src)
184         if (self%src(i)%par%id == srcid) then ! found the correct source to link to
185             call self%src(i)%linkproperty(ptype,pname) ! calling Source method to link property
186             temp = self%src(i)%par%id
187             outext='      Source id = '// temp // ', '// self%src(i)%par%name //' is of type '// self%src(i
   )%prop%property_type //', with property name ' // self%src(i)%prop%property_name
188             call log%put(outext,.false.)
189             notlinked = .false. ! we linked it
190             exit
191         endif
192     enddo
193     if (notlinked) then ! property has no corresponding Source
194         temp = srcid
195         outext='      Source id = '// temp //' not listed, property '// pname //', of type ' // ptype // '
  not linked, ignoring'
196         call log%put(outext,.false.)
197     endif
```

6.21.3 Variable Documentation

6.21.3.1 tempsources

```
type(source_group_class), public sources_mod::tempsources
```

Temporary Source array, used exclusively for building the case from a description file.

Definition at line 98 of file sources.f90.

```
98     type(source_group_class) :: tempSources
```

6.22 tracer_base_mod Module Reference

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

Data Types

- interface tracer
- 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 printtracer (self)

    Method to print basic info about the Tracer.
- type(tracer_class) function constructor (id, src, time, p)

    Base Tracer constructor.

Variables

- type(tracer_class), public dummytracer

    Just a template to allocate the generic arrays to this size.

6.22.1 Detailed Description

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

Author

Ricardo Birjukovs Canelas

6.22.2 Function/Subroutine Documentation

6.22.2.1 constructor()

```
type(tracer_class) function tracer_base_mod::constructor (
            integer, intent(in) id,
            class(source_class), intent(in) src,
            real(prec_time), intent(in) time,
            integer, intent(in) p )  [private]
```

Base Tracer constructor.

Author

Ricardo Birjukovs Canelas - MARETEC

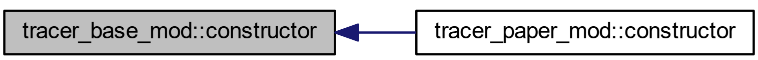
Parameters

in	*id,src,time,p*	

Definition at line 106 of file tracer_base.f90.

```
106     implicit none
107     type(tracer_class) :: constructor
108     integer, intent(in) :: id
109     class(source_class), intent(in) :: src
110     real(prec_time), intent(in) :: time
111     integer, intent(in) :: p
112
113     ! initialize parameters
114     constructor%par%id = id
115     constructor%par%idsource = src%par%id
116     constructor%par%velmax = 15.0 !(m/s, just a placeholder)
117     ! initialize tracer state
118     constructor%now%age=0.0
119     constructor%now%active = .true.
120     !print*, 'Source at'
121     !print*, src%now%pos
122     !print*, 'New tracer at'
123     !print*, src%stencil%ptlist(p) + src%now%pos
124     constructor%now%pos = src%stencil%ptlist(p) + src%now%pos
125     constructor%now%vel = 0.0
126     constructor%now%acc = 0.0
127     constructor%now%depth = 0.0
128     ! Initialize statistical accumulator variables
129     constructor%stats%acc_pos = 0.0
130     constructor%stats%acc_vel = 0.0
131     constructor%stats%acc_depth = 0.0
132     constructor%stats%ns = 0
133
```

Here is the caller graph for this function:



6.22.2.2 printtracer()

```
subroutine tracer_base_mod::printtracer (
            class(tracer_class), intent(inout) self )  [private]
```

Method to print basic info about the Tracer.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self*	

Definition at line 83 of file tracer_base.f90.

```
83      implicit none
84      class(tracer_class), intent(inout) :: self
85      type(string) :: outext, t(6)
86      if (self%now%active .eqv. .false.) then
87          outext = '-->Tracer is inactive'
88          call log%put(outext,.false.)
89      else
90          t(1) = self%par%id
91          t(2) = self%now%pos%x
92          t(3) = self%now%pos%y
93          t(4) = self%now%pos%z
94          outext = 'Tracer['//t(1)//']::xyz('//t(2)//','//t(3)//','//t(4)//')'
95          call log%put(outext,.false.)
96      end if
```

6.22.3 Variable Documentation

6.22.3.1 dummytracer

```
type(tracer_class), public tracer_base_mod::dummytracer
```

Just a template to allocate the generic arrays to this size.

Definition at line 62 of file tracer_base.f90.

```
62      type(tracer_class) :: dummyTracer
```

6.23 tracer_list_mod Module Reference

Module to hold the tracer linked list class and its methods. This class defines a double linked list to store any variable type, but with specific methods with type guards for Tracer objects. The class allows for insertion, deletion and iteration of the desired contents.

Data Types

- type tracerlist_class

Functions/Subroutines

- subroutine print_tracerlist (this)

  Method that prints all the links of the list.
- subroutine print_tracerlistcurrent (this)

  Method that prints the current link of the list.

6.23.1 Detailed Description

Module to hold the tracer linked list class and its methods. This class defines a double linked list to store any variable type, but with specific methods with type guards for Tracer objects. The class allows for insertion, deletion and iteration of the desired contents.

Author

Ricardo Birjukovs Canelas

6.23.2 Function/Subroutine Documentation

6.23.2.1 print_tracerlist()

```
subroutine tracer_list_mod::print_tracerlist (
            class(tracerlist_class), intent(in) this )  [private]
```

Method that prints all the links of the list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 47 of file tracer_list.f90.

```
47     class(tracerList_class), intent(in) :: this
48     class(*), pointer :: curr
49     call this%reset()               ! reset list iterator
50     do while(this%moreValues())     ! loop while there are values to print
51         call this%printCurrent()
52         call this%next()            ! increment the list iterator
53     end do
54     call this%reset()               ! reset list iterator
```

6.23.2.2 print_tracerlistcurrent()

```
subroutine tracer_list_mod::print_tracerlistcurrent (
             class(tracerlist_class), intent(in) this )  [private]
```

Method that prints the current link of the list.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 63 of file tracer_list.f90.

```
63      class(tracerList_class), intent(in) :: this
64      class(*), pointer :: curr
65      type(string) :: outext
66      curr => this%currentValue() ! get current value
67      select type(curr)
68      class is (tracer_class)
69          call curr%print()
70          class default
71          outext = '[tracerList_class::print] Unexepected type of content, not a Tracer'
72          call log%put(outext)
73          stop
74      end select
```

6.24 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.
- interface papertracer

Functions/Subroutines

- type(paper_class) function constructor (id, src, time, p)

    Paper Tracer constructor.

6.24.1 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

6.24.2 Function/Subroutine Documentation

6.24.2.1 constructor()

```
type(paper_class) function tracer_paper_mod::constructor (
            integer, intent(in) id,
            class(source_class), intent(in) src,
            real(prec_time), intent(in) time,
            integer, intent(in) p )  [private]
```

Paper Tracer constructor.

Author

Ricardo Birjukovs Canelas - MARETEC

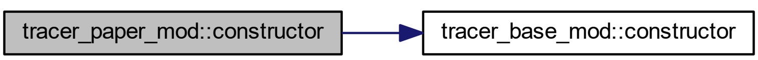
Parameters

in	*id,src,time,p*	

Definition at line 69 of file tracer_paper.f90.

```
69         implicit none
70         type(paper_class) :: constructor
71         integer, intent(in) :: id
72         class(source_class), intent(in) :: src
73         real(prec_time), intent(in) :: time
74         integer, intent(in) :: p
75         class(*), allocatable :: base_trc
76
77         !use the base class constructor to build the base of our new derived type
78         constructor%tracer_class = tracer(id,src,time,p)
79         !VERY NICE IFORT BUG (I think) - only some of the variables get used using the base constructor...
80         constructor%par%id = id !forcing
81         constructor%par%idsource = src%par%id !forcing
82         !now initialize the specific components of this derived type
83         !material parameters
84         constructor%mpar%degradation_rate = src%prop%degrd_rate
85         constructor%mpar%particulate = src%prop%particulate
86         constructor%mpar%size = src%prop%radius
87         !material state
88         constructor%mnow%density = src%prop%density
89         constructor%mnow%condition = src%prop%condition
90         constructor%mnow%radius = src%prop%radius
91         constructor%mnow%concentration = mv
92         if (constructor%mpar%particulate) then
93             constructor%mpar%size = src%prop%pt_radius !correcting size to now mean particle size, not tracer size
94             constructor%mnow%concentration = src%prop%ini_concentration
95         end if
96
```

Here is the call graph for this function:



6.25 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.

6.25.1 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

6.25.2 Function/Subroutine Documentation

6.25.2.1 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	*trc,id,id_source,time,pt*	

Definition at line 61 of file tracer_plastic.f90.

```
61      implicit none
62      class(plastic_class) :: trc
63      integer, intent(in) :: id
64      integer, intent(in) :: id_source
65      type(vector), intent(in) :: pt
66      real(prec_time), intent(in) :: time
67
68      ! initialize parameters
69      trc%par%id = id
70      trc%par%idsource = id_source
71      trc%par%velmax = 15.0 !(m/s, just a placeholder)
72      ! initialize tracer state
73      trc%now%age=0.0
74      trc%now%active = .false.
75      trc%now%pos = pt
76      trc%now%vel = 0.0
77      trc%now%acc = 0.0
78      trc%now%depth = 0.0
79      ! Initialize statistical accumulator variables
80      trc%stats%acc_pos = 0.0
81      trc%stats%acc_vel = 0.0
82      trc%stats%acc_depth = 0.0
83      trc%stats%ns = 0
84
```

6.26 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.

6.26.1 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

6.27 utilities_mod Module Reference

Module that provides useful back-end routines.

Functions/Subroutines

- type(vector) function, public geo2m (geovec, lat)

  Public function that returns a vector in meters given an array in geographical coordinates (lon, lat, z) and a lattitude.
- type(vector) function, public m2geo (mvec, lat)

  Public function that returns a vector in geographical coordinates (lon, lat, z) given an array in meters and a lattitude.
- character(:) function, allocatable, public int2str (fmt, i)

  Public function that returns a zero paded string from an integer number and a format descriptor.
- real(prec) function, public get_closest_twopow (num)

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

6.27.1 Detailed Description

Module that provides useful back-end routines.

Author

Ricardo Birjukovs Canelas

6.27.2 Function/Subroutine Documentation

6.27.2.1 geo2m()

```
type(vector) function, public utilities_mod::geo2m (
            type(vector), intent(in) geovec,
            real(prec), intent(in) lat )
```

Public function that returns a vector in meters given an array in geographical coordinates (lon, lat, z) and a lattitude.

Author

Ricardo Birjukovs Canelas - MARETEC

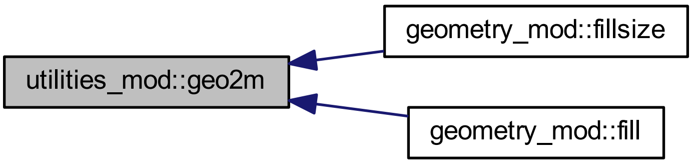
Parameters

in	*geovec,lat*	

Definition at line 43 of file utilities.f90.

```
43     type(vector), intent(in) :: geovec
44     real(prec), intent(in) :: lat
45     integer :: R
46     real(prec) :: pi
47     r = 6378137 !earth radius in meters
48     pi = 3.1415926
49     res = geovec
50     res%x = res%x*r*cos(pi*lat/180.0)
51     res%y = res%y*r
```

Here is the caller graph for this function:



6.27.2.2 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 96 of file utilities.f90.

```
96      implicit none
97      real(prec), intent(in) :: num
98      real(prec) :: twopow
99      integer :: i
100      real(prec) :: dist1, dist2
101      do i=-4, 10
102          twopow = 2.0**i
103          if (num < twopow) then
104              dist1 = sqrt(twopow-num)
105              dist2 = sqrt(num-2.0**(i-1))
106              if (dist2 < dist1) then
107                  twopow = 2.0**(i-1)
108                  exit
109              endif
110              exit
111          endif
112      enddo
```

6.27.2.3 int2str()

```
character(:)  function, allocatable, public utilities_mod::int2str (
            character(len=6), intent(in) fmt,
            integer, intent(in) i )
```

Public function that returns a zero paded string from an integer number and a format descriptor.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*fmt,i*	

Definition at line 81 of file utilities.f90.

```
81      character(:), allocatable :: res
82      character(len=6), intent(in) :: fmt ! format descriptor
83      integer, intent(in) :: i
84      character(range(i)+2) :: tmp
85      write(tmp, fmt) i
86      res = trim(tmp)
```

6.27.2.4 m2geo()

```
type(vector) function, public utilities_mod::m2geo (
            type(vector), intent(in) mvec,
            real(prec), intent(in) lat )
```

Public function that returns a vector in geographical coordinates (lon, lat, z) given an array in meters and a lattitude.

Author

Ricardo Birjukovs Canelas - MARETEC

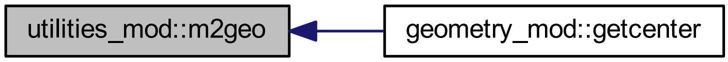
Parameters

in	*mvec,lat*	

Definition at line 62 of file utilities.f90.

```
62      type(vector), intent(in) :: mvec
63      real(prec), intent(in) :: lat
64      integer :: R
65      real(prec) :: pi
66      r = 6378137 !earth radius in meters
67      pi = 3.1415926
68      res = mvec
69      res%x = res%x/(r*cos(pi*lat/180.0))
70      res%y = res%y/r
```

Here is the caller graph for this function:



6.28 vtkwritter_mod Module Reference

Defines a vtk writer class with an object exposable to the Output streamer. Writes files in .xml vtk, both in serial and parallel model. Uses an unstructured mesh format specifier to store any type of data, both meshes and Tracers. Supports scalar and vectorial data.

Data Types

- type vtkwritter_class

Functions/Subroutines

- subroutine initvtkwritter (self)

    Initializes a VTK writer object.
- subroutine tracerserial (self, filename, blocks)

    Public Tracer writting routine. Writes Tracer data in binary XML VTK format using an unstructured grid. Serial writer for serial files.
- subroutine domain (self, filename, bbox, npbbox, blocks)

    Public simulation domain writting routine. Writes binary XML VTK format using an unstructured grid.

Variables

- type(vtkwritter_class), public vtkwritter

6.28.1 Detailed Description

Defines a vtk writer class with an object exposable to the Output streamer. Writes files in .xml vtk, both in serial and parallel model. Uses an unstructured mesh format specifier to store any type of data, both meshes and Tracers. Supports scalar and vectorial data.

Author

Ricardo Birjukovs Canelas

6.28.2 Function/Subroutine Documentation

6.28.2.1 domain()

```
subroutine vtkwritter_mod::domain (
            class(vtkwritter_class), intent(inout) self,
            type(string), intent(in) filename,
            class(boundingbox_class), intent(in) bbox,
            integer, intent(in) npbbox,
            class(block_class), dimension(:), intent(in) blocks ) [private]
```

Public simulation domain writting routine. Writes binary XML VTK format using an unstructured grid.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,filename,bbox,npbbox,blocks*	
in	*filename*	name of the case to add
in	*bbox*	Case bounding box
in	*blocks*	Case Blocks
in	*npbbox*	number of points of the bbox geometry

Definition at line 118 of file vtkwritter.f90.

```
118     implicit none
119     class(vtkwritter_class), intent(inout) :: self
120     type(string), intent(in) :: filename
121     class(boundingbox_class), intent(in) :: bbox
122     class(block_class), dimension(:), intent(in) :: blocks
123     integer, intent(in) :: npbbox
124
125     type(vtk_file) :: vtkfile
126     type(string) :: fullfilename
127     type(string) :: outext
128     integer :: error, i, b
129     integer, parameter :: nc = 1
130     real(prec), dimension(1:npbbox) :: xx, yy, zz
131     type(vector) :: pts(npbbox)
132     integer, dimension(1:npbbox) :: connect, var
133     integer(I4P), dimension(1:nc) :: offset
134     integer(I1P), dimension(1:nc) :: cell_type
135
136     offset = [8]
137     cell_type = [12]
138
139     !preparing file
140     fullfilename = filename%chars()//'_BoundingBox.vtu'
141     outext = '->Writting Bounding Box file '//fullfilename
142     call log%put(outext)
143     fullfilename = globals%Names%outpath//'/'//fullfilename
144
145     error = vtkfile%initialize(format=self%formatType%chars(), filename=fullfilename%chars(), mesh_topology
   ='UnstructuredGrid')
146
147     !Writting bounding box geometry
148     pts = geometry%getPoints(bbox)
149     do i=1, npbbox
150         xx(i) = pts(i)%x
```

```
        yy(i) = pts(i)%y
        zz(i) = pts(i)%z
        connect(i) = i-1
    end do
    error = vtkfile%xml_writer%write_piece(np=npbbox, nc=nc)
    error = vtkfile%xml_writer%write_geo(np=npbbox, nc=nc, x=xx, y=yy, z=zz)
    error = vtkfile%xml_writer%write_connectivity(nc=nc, connectivity=connect, offset=offset, cell_type=cell_type)
    error = vtkfile%xml_writer%write_piece()

    !Closing file
    error = vtkfile%finalize()

    !preparing file
    fullfilename = filename%chars()//'_Blocks.vtu'
    outext = '->Writting Blocks file '//fullfilename
    call log%put(outext)
    fullfilename = globals%Names%outpath//'/'//fullfilename

    error = vtkfile%initialize(format=self%formatType%chars(), filename=fullfilename%chars(), mesh_topology='UnstructuredGrid')

    !Writting block geometries
    do b=1, size(blocks)
        pts = geometry%getPoints(blocks(b)%extents)
        do i=1, npbbox
            xx(i) = pts(i)%x
            yy(i) = pts(i)%y
            !zz(i) = pts(i)%z
            connect(i) = i-1
            var(i) = b
        end do
        error = vtkfile%xml_writer%write_piece(np=npbbox, nc=nc)
        error = vtkfile%xml_writer%write_geo(np=npbbox, nc=nc, x=xx, y=yy, z=zz)
        error = vtkfile%xml_writer%write_connectivity(nc=nc, connectivity=connect, offset=offset, cell_type=cell_type)
        error = vtkfile%xml_writer%write_dataarray(location='node', action='open')
        error = vtkfile%xml_writer%write_dataarray(data_name='Block', x=var)
        error = vtkfile%xml_writer%write_dataarray(location='node', action='close')
        error = vtkfile%xml_writer%write_piece()
    end do

    !Closing file
    error = vtkfile%finalize()

```

6.28.2.2 initvtkwritter()

subroutine vtkwritter_mod::initvtkwritter (
class(vtkwritter_class), intent(inout) *self*) [private]

Initializes a VTK writer object.

Author

Ricardo Birjukovs Canelas - MARETEC

Definition at line 54 of file vtkwritter.f90.

```
54     implicit none
55     class(vtkwritter_class), intent(inout) :: self
56     self%numVtkFiles = 0
57     self%formatType = 'raw'
```

6.28.2.3 tracerserial()

```
subroutine vtkwritter_mod::tracerserial (
            class(vtkwritter_class), intent(inout) self,
            type(string), intent(in) filename,
            class(block_class), dimension(:), intent(in) blocks )  [private]
```

Public Tracer writting routine. Writes Tracer data in binary XML VTK format using an unstructured grid. Serial writer for serial files.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,filename,blocks*	
in	*blocks*	Case Blocks

Definition at line 68 of file vtkwritter.f90.

```
68      implicit none
69      class(vtkwritter_class), intent(inout) :: self
70      type(string), intent(in) :: filename
71      class(block_class), dimension(:), intent(in) :: blocks
72
73      type(vtk_file) :: vtkfile
74      type(string) :: fullfilename
75      type(string) :: outext
76      integer :: error, i
77      integer :: np
78      integer, parameter :: nc = 0
79      integer(I1P), dimension(1:nc) :: cell_type
80      integer(I4P), dimension(1:nc) :: offset
81      integer(I4P), dimension(:), allocatable :: connect
82
83      fullfilename = filename%chars()//'.vtu'
84      outext = '->Writting output file '//fullfilename
85      call log%put(outext)
86      fullfilename = globals%Names%outpath//'/'//fullfilename
87
88      error = vtkfile%initialize(format=self%formatType%chars(), filename=fullfilename%chars(), mesh_topology='UnstructuredGrid')
89      !Write the data of each block
90      do i = 1, size(blocks)
91          if (blocks(i)%LTracer%getSize() > 0) then
92              np = blocks(i)%LTracer%getSize()
93              allocate(connect(np))
94              error = vtkfile%xml_writer%write_piece(np=np, nc=nc)
95              error = vtkfile%xml_writer%write_geo(np=np, nc=nc, x=blocks(i)%AoT%x, y=blocks(i)%AoT%y, z=blocks(i)%AoT%z)
96              error = vtkfile%xml_writer%write_connectivity(nc=nc, connectivity=connect, offset=offset, cell_type=cell_type)
97              error = vtkfile%xml_writer%write_dataarray(location='node', action='open')
98              error = vtkfile%xml_writer%write_dataarray(data_name='id', x=blocks(i)%AoT%id)
99              error = vtkfile%xml_writer%write_dataarray(data_name='velocity', x=blocks(i)%AoT%u, y=blocks(i)%AoT%v, z=blocks(i)%AoT%w)
100             error = vtkfile%xml_writer%write_dataarray(location='node', action='close')
101             error = vtkfile%xml_writer%write_piece()
102             deallocate(connect)
103         end if
104     end do
105     error = vtkfile%finalize()
106     self%numVtkFiles = self%numVtkFiles + 1
107
```

6.28.3 Variable Documentation

6.28.3.1 vtkwritter

```
type(vtkwritter_class), public vtkwritter_mod::vtkwritter
```

Definition at line 41 of file vtkwritter.f90.

```
41      type(vtkwritter_class) :: vtkWritter
```

6.29 xmlparser_mod Module Reference

Module with the simulation xml parsing class and methods, Encapsulates the FOX_dom library.

Data Types

- type xmlparser_class

Functions/Subroutines

- subroutine getfile (self, xmldoc, xmlfilename)

    Method that parses an xml file and returns a pointer to the master node.
- subroutine closefile (self, xmldoc)

    Method that closes a parsed xml file or node.
- subroutine getleafattribute (self, xmlnode, att_name, att_value)

    Method that parses an xml attribute. Reads the requested attribute from a given leaf node,.
- subroutine getnodeattribute (self, xmlnode, tag, att_name, att_value, read_flag, mandatory)

    Method that parses an attribute from an xml node. In the format '<Tag att_name="att_value">'.
- subroutine getnodevector (self, xmlnode, tag, vec, read_flag, mandatory)

    Method to parse xyz vectors in xml files. Vector must be in format '<Tag x="vec%x" y="vec%y" z="vec%z">'.
- subroutine gotonode (self, currentNode, targetNode, targetNodeName, read_flag, mandatory)

    Method that retrieves a node from within a node. Returns a nullifyed pointer if not found, stops if mandatory.

Variables

- type(xmlparser_class), public xmlreader

6.29.1 Detailed Description

Module with the simulation xml parsing class and methods, Encapsulates the FOX_dom library.

Author

Ricardo Birjukovs Canelas

6.29.2 Function/Subroutine Documentation

6.29.2.1 closefile()

```
subroutine xmlparser_mod::closefile (
            class(xmlparser_class), intent(in) self,
            type(node), intent(out), pointer xmldoc )  [private]
```

Method that closes a parsed xml file or node.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,xmldoc*	
out	*xmldoc*	Node that conatins the parsed file

Definition at line 78 of file xmlparser.f90.

```
78      implicit none
79      class(xmlparser_class), intent(in) :: self
80      type(Node), intent(out), pointer :: xmldoc
81      call destroy(xmldoc) !using FOX function
```

6.29.2.2 getfile()

```
subroutine xmlparser_mod::getfile (
            class(xmlparser_class), intent(in) self,
            type(node), intent(out), pointer xmldoc,
            type(string), intent(in) xmlfilename )  [private]
```

Method that parses an xml file and returns a pointer to the master node.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,xmldoc,xmlfilename*	
out	*xmldoc*	Node that conatins the parsed file
in	*xmlfilename*	File name

Definition at line 54 of file xmlparser.f90.

```
54      implicit none
55      class(xmlparser_class), intent(in) :: self
56      type(Node), intent(out), pointer :: xmldoc
```

```
    type(string), intent(in) :: xmlfilename
    integer :: err
    type(string) :: outext
    xmldoc => parsefile(xmlfilename%chars(), iostat=err) !using FOX function
    if (err==0) then
        outext='->Reading .xml file '//xmlfilename
        call log%put(outext)
    else
        outext='[XMLReader::getFile]: no '//xmlfilename//' file, or file is invalid. Stoping'
        call log%put(outext)
        stop
    endif
```

6.29.2.3 getleafattribute()

```
subroutine xmlparser_mod::getleafattribute (
            class(xmlparser_class), intent(in) self,
            type(node), intent(in), pointer xmlnode,
            type(string), intent(in) att_name,
            type(string), intent(out) att_value )  [private]
```

Method that parses an xml attribute. Reads the requested attribute from a given leaf node,.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,xmlnode,att_name,att_value*	
in	*xmlnode*	Working xml node
in	*att_name*	Atribute name to collect from tag
out	*att_value*	Attribute value

Definition at line 92 of file xmlparser.f90.

```
92     implicit none
93     class(xmlparser_class), intent(in) :: self
94     type(Node), intent(in), pointer :: xmlnode
95     type(string), intent(in) :: att_name
96     type(string), intent(out) :: att_value
97     character(80) :: att_value_chars
98     call extractdataattribute(xmlnode, att_name%chars(), att_value_chars) !using FOX function
99     att_value=trim(att_value_chars)
```

6.29.2.4 getnodeattribute()

```
subroutine xmlparser_mod::getnodeattribute (
            class(xmlparser_class), intent(in) self,
            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]
```

Method that parses an attribute from an xml node. In the format '<Tag att_name="att_value">'.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,xmlnode,tag,att_name,att_value,read_flag,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 110 of file xmlparser.f90.

```
110     implicit none
111     class(xmlparser_class), intent(in) :: self
112     type(Node), intent(in), pointer :: xmlnode
113     type(string), intent(in) :: tag
114     type(string), intent(in) :: att_name
115     type(string), intent(out) :: att_value
116     logical, intent(out), optional :: read_flag
117     logical, intent(in), optional :: mandatory
118
119     type(string) :: outext, nodename
120     character(80) :: att_value_chars
121     type(NodeList), pointer :: target_node_list, nodeChildren
122     type(Node), pointer :: nodedetail
123     logical :: validtag
124     integer :: i
125
126     validtag = .false.
127     nodechildren => getchildnodes(xmlnode) !getting all of the nodes bellow the main source node (all of it's private info) !using FOX function
128     do i=0, getlength(nodechildren)-1
129         nodedetail => item(nodechildren,i) !grabing a node !using FOX function
130         nodename = getlocalname(nodedetail)  !finding its name !using FOX function
131         if (nodename == tag) then
132             validtag=.true.
133             exit
134         endif
135     enddo
136     if (validtag) then
137         target_node_list => getelementsbytagname(xmlnode, tag%chars())   !searching for tags with the given name !using FOX function
138         nodedetail => item(target_node_list, 0) !using FOX function
139         call extractdataattribute(nodedetail, att_name%chars(), att_value_chars) !using FOX function
140         att_value=trim(att_value_chars)
141         if (present(read_flag)) then
142             read_flag =.true.
143         endif
144     else
145         if(present(mandatory)) then
146             if(mandatory.eqv..false.) then
147                 if (present(read_flag)) then
148                     read_flag =.false.
149                 endif
150             endif
151         else
152             outext='Could not find any "'//tag//'" tag for xml node "'//getnodename(xmlnode)//'", stoping'
153             call log%put(outext)
154             stop
155         endif
156     endif
```

6.29.2.5 getnodevector()

```
subroutine xmlparser_mod::getnodevector (
            class(xmlparser_class), intent(in) self,
            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]
```

Method to parse xyz vectors in xml files. Vector must be in format '<Tag x="vec%x" y="vec%y" z="vec%z">'.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

in	*self,xmlnode,tag,vec,read_flag,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 167 of file xmlparser.f90.

```
167     implicit none
168     class(xmlparser_class), intent(in) :: self
169     type(Node), intent(in), pointer :: xmlnode
170     type(string), intent(in) :: tag
171     type(vector), intent(out) :: vec
172     logical, intent(out), optional :: read_flag
173     logical, intent(in), optional :: mandatory
174
175     type(string) :: outext, nodename
176     type(NodeList), pointer :: target_node_list, nodeChildren
177     type(Node), pointer :: nodedetail
178     logical :: validtag
179     integer :: i
180
181     vec%x=mv !marking the array as not read
182     validtag = .false.
183     nodechildren => getchildnodes(xmlnode) !getting all of the nodes bellow the main source node (all of it's private info) !using FOX function
184     do i=0, getlength(nodechildren)-1
185         nodedetail => item(nodechildren,i) !grabing a node !using FOX function
186         nodename = getlocalname(nodedetail)  !finding its name !using FOX function
187         if (nodename == tag) then
188             validtag =.true.
189             exit
190         endif
191     enddo
192     if (validtag) then
193         target_node_list => getelementsbytagname(xmlnode, tag%chars())   !searching for tags with the given name !using FOX function
194         nodedetail => item(target_node_list, 0) !using FOX function
195         call extractdataattribute(nodedetail, "x", vec%x) !using FOX function
196         call extractdataattribute(nodedetail, "y", vec%y)
197         call extractdataattribute(nodedetail, "z", vec%z)
198         if (present(read_flag)) then
199             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
```

6.29.2.6 gotonode()

```
subroutine xmlparser_mod::gotonode (
            class(xmlparser_class), intent(in) self,
            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]
```

Method that retrieves a node from within a node. Returns a nullifyed pointer if not found, stops if mandatory.

Author

Ricardo Birjukovs Canelas - MARETEC

Parameters

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

Definition at line 224 of file xmlparser.f90.

```
    implicit none
    class(xmlparser_class), intent(in) :: self
    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

    type(NodeList), pointer :: target_node_list
    type(string) :: outext, nodename
    integer :: i
    logical :: target_node_exists

    target_node_exists = .false.
    target_node_list => getchildnodes(currentnode) !using FOX function
    do i=0, getlength(target_node_list)-1
        targetnode => item(target_node_list,i) !grabing a node !using FOX function
        nodename = getlocalname(targetnode)  !finding its name !using FOX function
        if (nodename == targetnodename) then !found our target node
            target_node_exists = .true.
            if (present(read_flag)) then
```

```
                read_flag =.true.
            endif
            exit
        endif
    enddo
    if (target_node_exists .eqv. .false.) then
        nullify(targetnode)
        if(present(mandatory)) then
            if (mandatory.eqv..false.) then
                outext='Could not find any node called "'//targetnodename//'" in the xml file, ignoring'
                call log%put(outext)
                if (present(read_flag)) then
                    read_flag =.false.
                endif
            else
                outext='Could not find any node called "'//targetnodename//'" in the xml file, stoping'
                call log%put(outext)
                stop
            endif
        else
            outext='Could not find any node called "'//targetnodename//'" in the xml file, stoping'
            call log%put(outext)
            stop
        endif
    endif
```

6.29.3 Variable Documentation

6.29.3.1 xmlreader

`type(xmlparser_class), public xmlparser_mod::xmlreader`

Definition at line 40 of file xmlparser.f90.

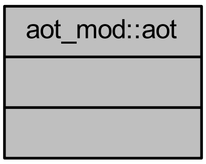
```
    type(xmlparser_class) :: XMLReader
```

Chapter 7

Data Type Documentation

7.1 aot_mod::aot Interface Reference

Collaboration diagram for aot_mod::aot:



7.1.1 Detailed Description

Definition at line 47 of file AoT.f90.

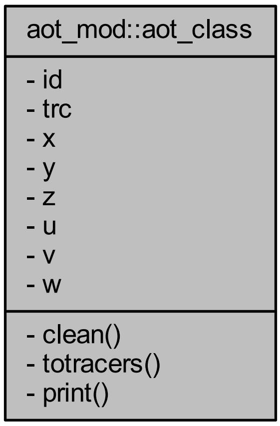
The documentation for this interface was generated from the following file:

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

7.2 aot_mod::aot_class Type Reference

Arrays of Tracers class.

Collaboration diagram for aot_mod::aot_class:



Private Member Functions

- procedure clean
- procedure totracers
- procedure print => print_AoT

Private Attributes

- integer, dimension(:), allocatable id

  Id of the Tracer.
- class(trc_ptr_class), dimension(:), allocatable trc

  pointer to the Tracer
- real(prec), dimension(:), allocatable x
- real(prec), dimension(:), allocatable y
- real(prec), dimension(:), allocatable z

  coordinates of the Tracer
- real(prec), dimension(:), allocatable u
- real(prec), dimension(:), allocatable v
- real(prec), dimension(:), allocatable w

  velocities of the Tracer

7.2.1 Detailed Description

Arrays of Tracers class.

Definition at line 35 of file AoT.f90.

7.2.2 Member Function/Subroutine Documentation

7.2.2.1 clean()

```
procedure aot_mod::aot_class::clean ( )  [private]
```

Definition at line 42 of file AoT.f90.

7.2.2.2 print()

```
procedure aot_mod::aot_class::print ( )  [private]
```

Definition at line 44 of file AoT.f90.

7.2.2.3 totracers()

```
procedure aot_mod::aot_class::totracers ( )  [private]
```

Definition at line 43 of file AoT.f90.

7.2.3 Member Data Documentation

7.2.3.1 id

```
integer, dimension(:), allocatable aot_mod::aot_class::id  [private]
```

Id of the Tracer.

Definition at line 36 of file AoT.f90.

```
36         integer, allocatable, dimension(:) :: id
```

7.2.3.2 trc

`class(trc_ptr_class), dimension(:), allocatable aot_mod::aot_class::trc [private]`

pointer to the Tracer

Definition at line 37 of file AoT.f90.

```
37         class(trc_ptr_class), allocatable, dimension(:) :: trc
```

7.2.3.3 u

`real(prec), dimension(:), allocatable aot_mod::aot_class::u [private]`

Definition at line 39 of file AoT.f90.

```
39         real(prec), allocatable, dimension(:) :: u,v,w
```

7.2.3.4 v

`real(prec), dimension(:), allocatable aot_mod::aot_class::v [private]`

Definition at line 39 of file AoT.f90.

7.2.3.5 w

`real(prec), dimension(:), allocatable aot_mod::aot_class::w [private]`

velocities of the Tracer

Definition at line 39 of file AoT.f90.

7.2.3.6 x

`real(prec), dimension(:), allocatable aot_mod::aot_class::x [private]`

Definition at line 38 of file AoT.f90.

```
38         real(prec), allocatable, dimension(:) :: x,y,z
```

7.2.3.7 y

```
real(prec), dimension(:), allocatable aot_mod::aot_class::y  [private]
```

Definition at line 38 of file AoT.f90.

7.2.3.8 z

```
real(prec), dimension(:), allocatable aot_mod::aot_class::z  [private]
```

coordinates of the Tracer

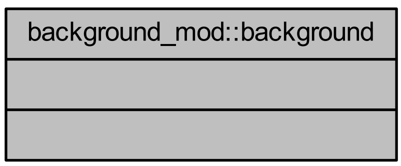
Definition at line 38 of file AoT.f90.

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

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

7.3 background_mod::background Interface Reference

Collaboration diagram for background_mod::background:



7.3.1 Detailed Description

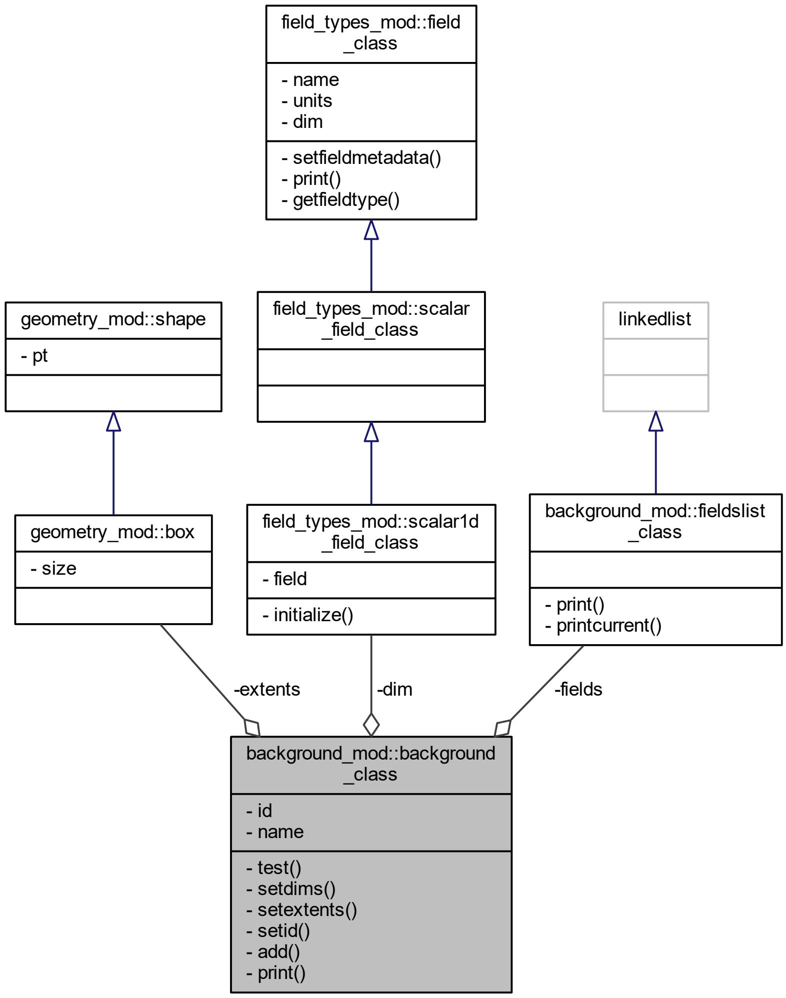
Definition at line 52 of file background.f90.

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

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

7.4 background_mod::background_class Type Reference

Collaboration diagram for background_mod::background_class:



Private Member Functions

- procedure test
- procedure, private setdims
- procedure, private setextents
- procedure, private setid
- procedure add => addField
- procedure print => printBackground

Private Attributes

- integer id = 0

ID of the Background.

- type(string) name

  Name of the Background.

- type(box) extents

  shape::box that defines the extents of the Background solution

- type(scalar1d_field_class), dimension(:), allocatable dim

  Dimensions of the Background fields (time,lon,lat,depth for example)

- type(fieldslist_class) fields

  Linked list to store the fields in the Background.

7.4.1 Detailed Description

Definition at line 37 of file background.f90.

7.4.2 Member Function/Subroutine Documentation

7.4.2.1 add()

```
procedure background_mod::background_class::add ( )  [private]
```

Definition at line 48 of file background.f90.

7.4.2.2 print()

```
procedure background_mod::background_class::print ( )  [private]
```

Definition at line 49 of file background.f90.

7.4.2.3 setdims()

```
procedure, private background_mod::background_class::setdims ( )  [private]
```

Definition at line 45 of file background.f90.

7.4.2.4 setextents()

```
procedure, private background_mod::background_class::setextents ( )  [private]
```

Definition at line 46 of file background.f90.

7.4.2.5 setid()

```
procedure, private background_mod::background_class::setid ( ) [private]
```

Definition at line 47 of file background.f90.

7.4.2.6 test()

```
procedure background_mod::background_class::test ( ) [private]
```

Definition at line 44 of file background.f90.

7.4.3 Member Data Documentation

7.4.3.1 dim

```
type(scalar1d_field_class), dimension(:), allocatable background_mod::background_class::dim
[private]
```

Dimensions of the Background fields (time,lon,lat,depth for example)

Definition at line 41 of file background.f90.

```
41         type(scalar1d_field_class), allocatable, dimension(:) :: dim
```

7.4.3.2 extents

```
type(box) background_mod::background_class::extents [private]
```

shape::box that defines the extents of the Background solution

Definition at line 40 of file background.f90.

```
40         type(box) :: extents
```

7.4.3.3 fields

```
type(fieldslist_class) background_mod::background_class::fields  [private]
```

Linked list to store the fields in the Background.

Definition at line 42 of file background.f90.

```
42         type(fieldsList_class) :: fields
```

7.4.3.4 id

```
integer background_mod::background_class::id = 0  [private]
```

ID of the Background.

Definition at line 38 of file background.f90.

```
38         integer :: id = 0
```

7.4.3.5 name

```
type(string) background_mod::background_class::name  [private]
```

Name of the Background.

Definition at line 39 of file background.f90.

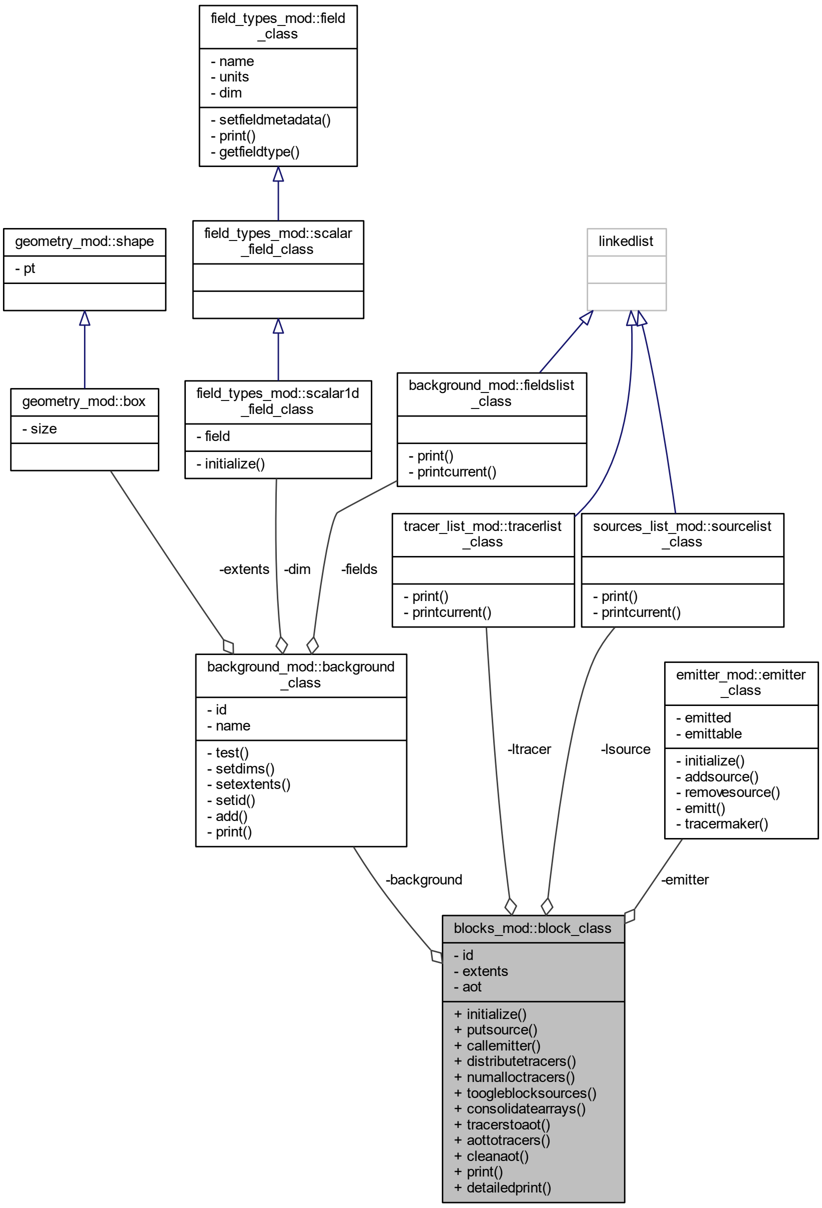
```
39         type(string) :: name
```

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

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

7.5 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 callemitter
- procedure, public distributetracers
- procedure, public numalloctracers
- procedure, public toogleblocksources

- procedure, public consolidatearrays
- procedure, public tracerstoaot
- procedure, public aottotracers
- procedure, public cleanaot
- 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(sourcelist_class) lsource

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

  Block Emitter.
- type(tracerlist_class) ltracer

  List of Tracers currently on this block.
- type(aot_class) aot

  Block Array of Tracers for actual numerical work.
- type(background_class), dimension(:), allocatable background

  Solution Backgrounds for the Block.

7.5.1 Detailed Description

Definition at line 38 of file blocks.f90.

7.5.2 Member Function/Subroutine Documentation

7.5.2.1 aottotracers()

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

Definition at line 56 of file blocks.f90.

7.5.2.2 callemitter()

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

Definition at line 50 of file blocks.f90.

7.5.2.3 cleanaot()

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

Definition at line 57 of file blocks.f90.

7.5.2.4 consolidatearrays()

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

Definition at line 54 of file blocks.f90.

7.5.2.5 detailedprint()

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

Definition at line 59 of file blocks.f90.

7.5.2.6 distributetracers()

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

Definition at line 51 of file blocks.f90.

7.5.2.7 initialize()

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

Definition at line 48 of file blocks.f90.

7.5.2.8 numalloctracers()

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

Definition at line 52 of file blocks.f90.

7.5.2.9 print()

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

Definition at line 58 of file blocks.f90.

7.5.2.10 putsource()

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

Definition at line 49 of file blocks.f90.

7.5.2.11 toogleblocksources()

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

Definition at line 53 of file blocks.f90.

7.5.2.12 tracerstoaot()

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

Definition at line 55 of file blocks.f90.

7.5.3 Member Data Documentation

7.5.3.1 aot

```
type(aot_class) blocks_mod::block_class::aot  [private]
```

Block Array of Tracers for actual numerical work.

Definition at line 44 of file blocks.f90.

```
44        type(aot_class)        :: AoT
```

7.5.3.2 background

`type(background_class), dimension(:), allocatable blocks_mod::block_class::background [private]`

Solution Backgrounds for the Block.

Definition at line 45 of file blocks.f90.

```
45         type(background_class), allocatable, dimension(:) :: Background
```

7.5.3.3 emitter

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

Block Emitter.

Definition at line 42 of file blocks.f90.

```
42         type(emitter_class)    :: Emitter
```

7.5.3.4 extents

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

shape::box that defines the extents of this block

Definition at line 40 of file blocks.f90.

```
40         type(box) :: extents
```

7.5.3.5 id

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

Definition at line 39 of file blocks.f90.

```
39         integer :: id
```

7.5.3.6 lsource

`type(sourcelist_class) blocks_mod::block_class::lsource [private]`

List of Sources currently on this block.

Definition at line 41 of file blocks.f90.

```
41         type(sourceList_class) :: LSource
```

7.5.3.7 ltracer

`type(tracerlist_class) blocks_mod::block_class::ltracer [private]`

List of Tracers currently on this block.

Definition at line 43 of file blocks.f90.

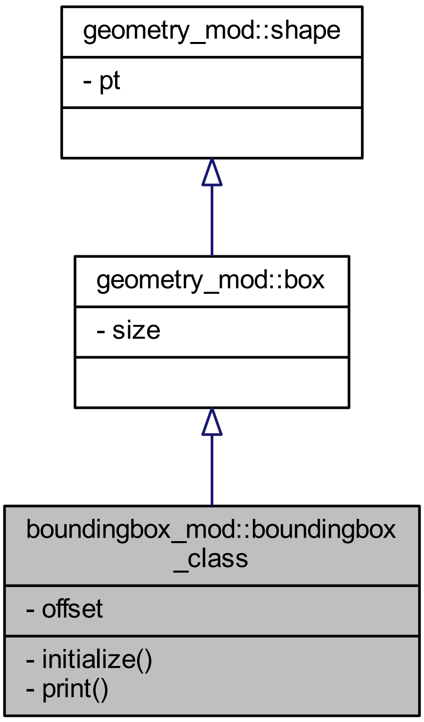
```
43         type(tracerList_class) :: LTracer
```

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

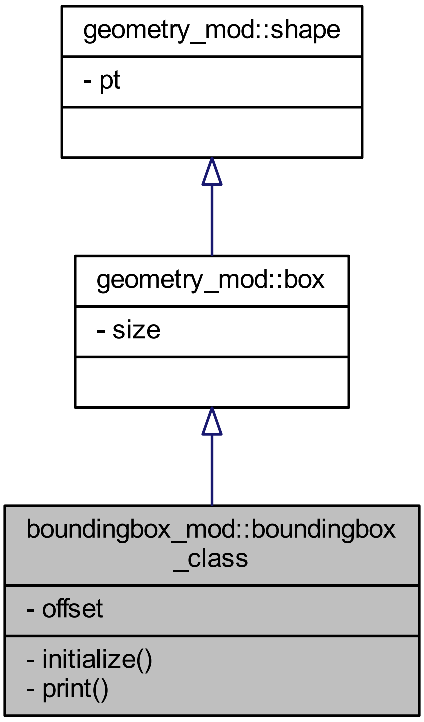
- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/blocks.f90

7.6 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

7.6.1 Detailed Description

Definition at line 26 of file boundingbox.f90.

7.6.2 Member Function/Subroutine Documentation

7.6.2.1 initialize()

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

Definition at line 29 of file boundingbox.f90.

7.6.2.2 print()

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

Definition at line 30 of file boundingbox.f90.

7.6.3 Member Data Documentation

7.6.3.1 offset

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

Definition at line 27 of file boundingbox.f90.

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

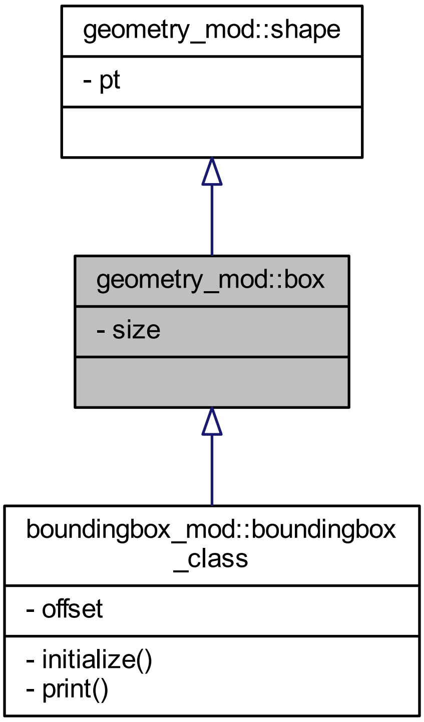
The documentation for this type was generated from the following file:

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

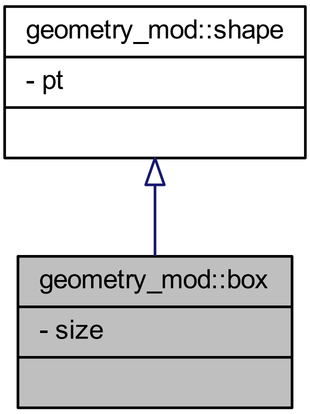
7.7 geometry_mod::box Type Reference

Type - point class.

Inheritance diagram for geometry_mod::box:



Collaboration diagram for geometry_mod::box:



Private Attributes

- type(vector) size
  Box size (m)

7.7.1 Detailed Description

Type - point class.

Definition at line 61 of file geometry.f90.

7.7.2 Member Data Documentation

7.7.2.1 size

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

Box size (m)

Definition at line 62 of file geometry.f90.

```
62        type(vector) :: size
```

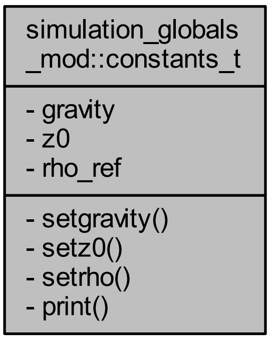
The documentation for this type was generated from the following file:

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

7.8 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)

7.8.1 Detailed Description

Case Constants class.

Definition at line 70 of file simulation_globals.f90.

7.8.2 Member Function/Subroutine Documentation

7.8.2.1 print()

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

Definition at line 78 of file simulation_globals.f90.

7.8.2.2 setgravity()

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

Definition at line 75 of file simulation_globals.f90.

7.8.2.3 setrho()

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

Definition at line 77 of file simulation_globals.f90.

7.8.2.4 setz0()

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

Definition at line 76 of file simulation_globals.f90.

7.8.3 Member Data Documentation

7.8.3.1 gravity

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

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

Definition at line 71 of file simulation_globals.f90.

```
71         type(vector) :: Gravity
```

7.8.3.2 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 73 of file simulation_globals.f90.

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

7.8.3.3 z0

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

Reference local sea level.

Definition at line 72 of file simulation_globals.f90.

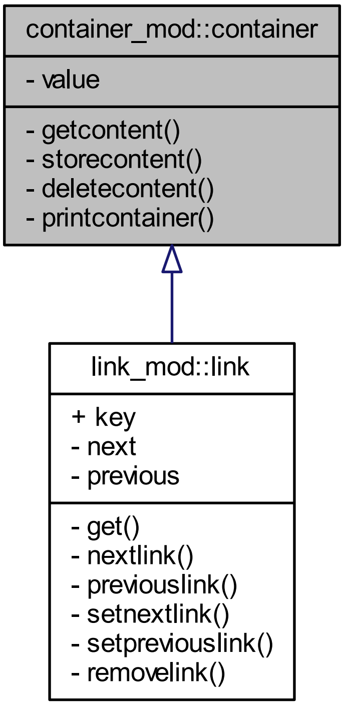
```
72         real(prec)    :: Z0 = 0.0
```

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

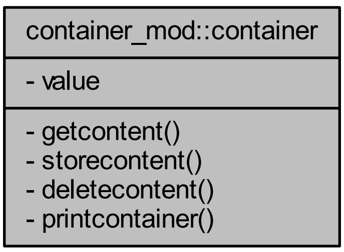
- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/simulation_globals.f90

7.9 container_mod::container Interface Reference

Inheritance diagram for container_mod::container:



Collaboration diagram for container_mod::container:



Private Member Functions

- procedure getcontent

returns stored content (pointer)
- procedure storecontent
  stores the provided values (sourced allocation)
- procedure deletecontent
  deletes the content of the container
- procedure printcontainer
  prints container contents (only primitive types implemented)

Private Attributes

- class(∗), pointer value => null()
  value stored in container

7.9.1 Detailed Description

Definition at line 40 of file container.f90.

7.9.2 Member Function/Subroutine Documentation

7.9.2.1 deletecontent()

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

deletes the content of the container

Definition at line 46 of file container.f90.

7.9.2.2 getcontent()

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

returns stored content (pointer)

Definition at line 44 of file container.f90.

7.9.2.3 printcontainer()

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

prints container contents (only primitive types implemented)

Definition at line 47 of file container.f90.

7.9.2.4 storecontent()

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

stores the provided values (sourced allocation)

Definition at line 45 of file container.f90.

7.9.3 Member Data Documentation

7.9.3.1 value

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

value stored in container

Definition at line 42 of file container.f90.

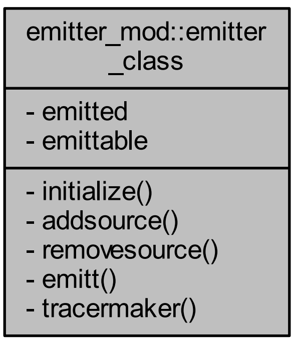
```
42        class(*), pointer :: value => null()
```

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

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

7.10 emitter_mod::emitter_class Type Reference

Collaboration diagram for emitter_mod::emitter_class:



Private Member Functions

- procedure initialize => initializeEmitter
- procedure addsource
- procedure removesource
- procedure emitt
- procedure tracermaker

Private Attributes

- integer emitted

    number of Tracers this Emitter has created
- integer emittable

    number of Tracers this Emitter should create throughout the simulation

7.10.1 Detailed Description

Definition at line 32 of file emitter.f90.

7.10.2 Member Function/Subroutine Documentation

7.10.2.1 addsource()

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

Definition at line 37 of file emitter.f90.

7.10.2.2 emitt()

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

Definition at line 39 of file emitter.f90.

7.10.2.3 initialize()

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

Definition at line 36 of file emitter.f90.

7.10.2.4 removesource()

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

Definition at line 38 of file emitter.f90.

7.10.2.5 tracermaker()

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

Definition at line 40 of file emitter.f90.

7.10.3 Member Data Documentation

7.10.3.1 emittable

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

number of Tracers this Emitter should create throughout the simulation

Definition at line 34 of file emitter.f90.

```
34        integer :: emittable
```

7.10.3.2 emitted

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

number of Tracers this Emitter has created

Definition at line 33 of file emitter.f90.

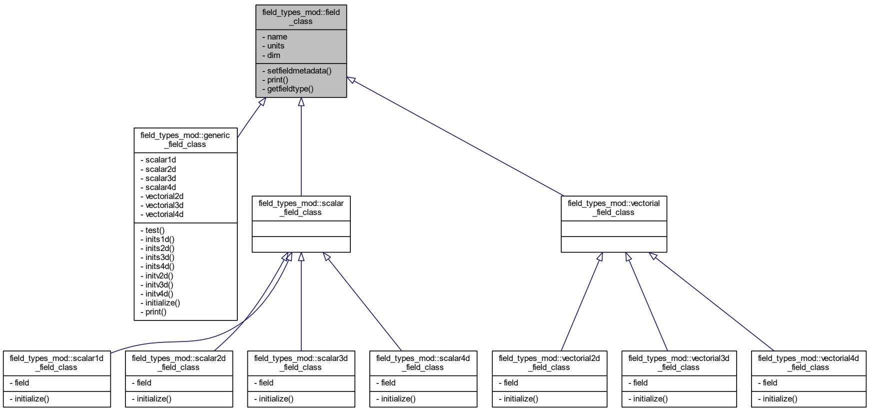
```
33        integer :: emitted
```

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

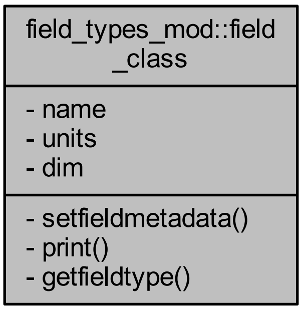
- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/emitter.f90

7.11 field_types_mod::field_class Type Reference

Inheritance diagram for field_types_mod::field_class:



Collaboration diagram for field_types_mod::field_class:



Private Member Functions

- procedure setfieldmetadata
- procedure print => printField

  Method that prints the field information.
- procedure getfieldtype

  Method that returns the field type (scalar or vectorial), in a string.

Private Attributes

- type(string) name

  name of the field
- type(string) units

  units of the field, preferably SI please
- integer dim

  dimensions of the field (1, 2, 3 or 4D)

7.11.1 Detailed Description

Definition at line 31 of file fields_types.f90.

7.11.2 Member Function/Subroutine Documentation

7.11.2.1 getfieldtype()

`procedure field_types_mod::field_class::getfieldtype ( )  [private]`

Method that returns the field type (scalar or vectorial), in a string.

Definition at line 38 of file fields_types.f90.

7.11.2.2 print()

`procedure field_types_mod::field_class::print ( )  [private]`

Method that prints the field information.

Definition at line 37 of file fields_types.f90.

7.11.2.3 setfieldmetadata()

`procedure field_types_mod::field_class::setfieldmetadata ( )  [private]`

Definition at line 36 of file fields_types.f90.

7.11.3 Member Data Documentation

7.11.3.1 dim

`integer field_types_mod::field_class::dim  [private]`

dimensions of the field (1, 2, 3 or 4D)

Definition at line 34 of file fields_types.f90.

```
34        integer :: dim
```

7.11.3.2 name

```
type(string) field_types_mod::field_class::name  [private]
```

name of the field

Definition at line 32 of file fields_types.f90.

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

7.11.3.3 units

```
type(string) field_types_mod::field_class::units  [private]
```

units of the field, preferably SI please

Definition at line 33 of file fields_types.f90.

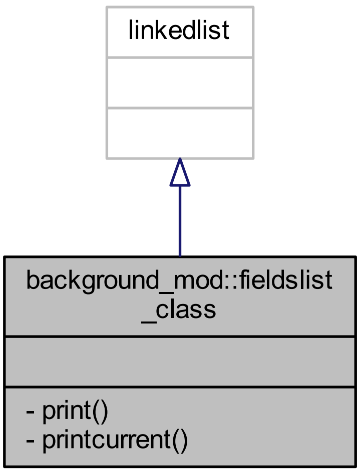
```
33         type(string) :: units
```

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

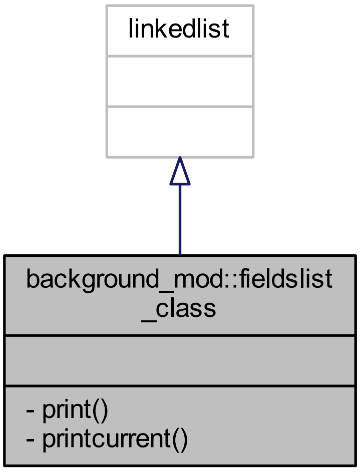
- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/fields_types.f90

7.12 background_mod::fieldslist_class Type Reference

Inheritance diagram for background_mod::fieldslist_class:



Collaboration diagram for background_mod::fieldslist_class:



Private Member Functions

- procedure print => print_fieldList
- procedure printcurrent => print_fieldListCurrent

7.12.1 Detailed Description

Definition at line 31 of file background.f90.

7.12.2 Member Function/Subroutine Documentation

7.12.2.1 print()

`procedure background_mod::fieldslist_class::print ( ) [private]`

Definition at line 33 of file background.f90.

7.12.2.2 printcurrent()

```
procedure background_mod::fieldslist_class::printcurrent ( )  [private]
```

Definition at line 34 of file background.f90.

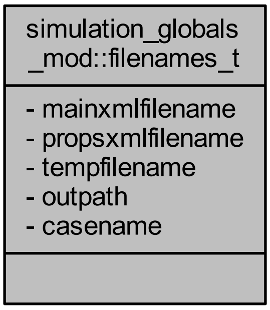
The documentation for this type was generated from the following file:

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

7.13 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.
- type(string) outpath

  General output directory.
- type(string) casename

  Name of the running case.

7.13.1 Detailed Description

File names class.

Definition at line 81 of file simulation_globals.f90.

7.13.2 Member Data Documentation

7.13.2.1 casename

type(string) simulation_globals_mod::filenames_t::casename [private]

Name of the running case.

Definition at line 86 of file simulation_globals.f90.

```
86        type(string) :: casename
```

7.13.2.2 mainxmlfilename

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

Input .xml file name.

Definition at line 82 of file simulation_globals.f90.

```
82        type(string) :: mainxmlfilename
```

7.13.2.3 outpath

type(string) simulation_globals_mod::filenames_t::outpath [private]

General output directory.

Definition at line 85 of file simulation_globals.f90.

```
85        type(string) :: outpath
```

7.13.2.4 propsxmlfilename

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

Properties .xml file name.

Definition at line 83 of file simulation_globals.f90.

```
83        type(string) :: propsxmlfilename
```

7.13.2.5 tempfilename

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

Generic temporary file name.

Definition at line 84 of file simulation_globals.f90.

```
84         type(string) :: tempfilename
```

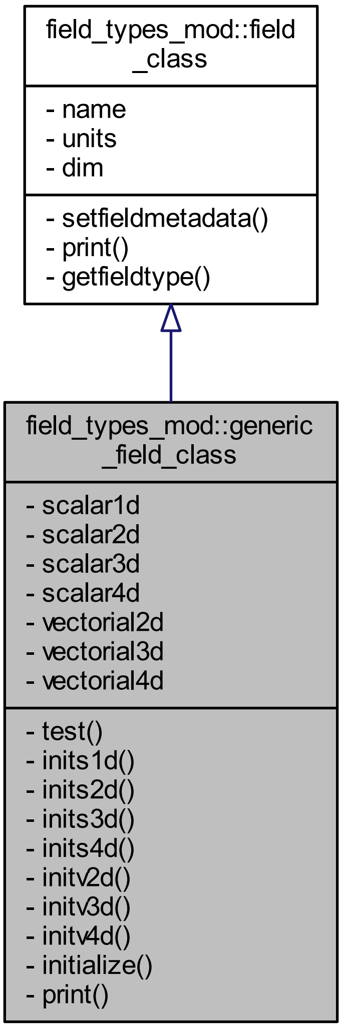
The documentation for this type was generated from the following file:

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

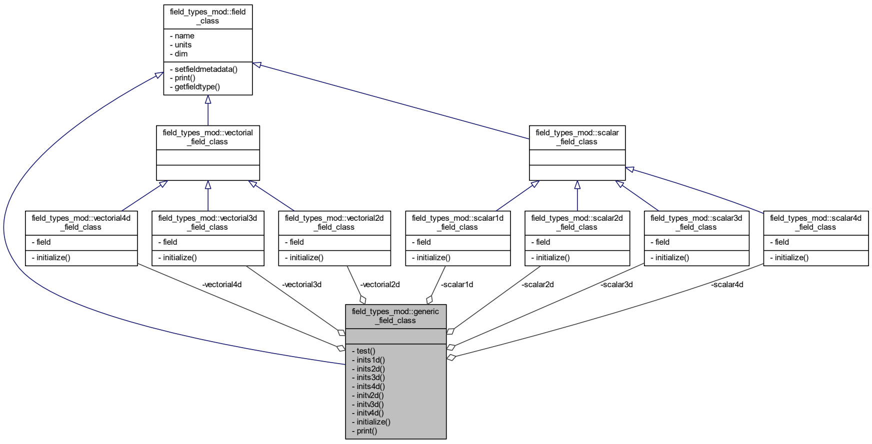
7.14 field_types_mod::generic_field_class Type Reference

generic field class. This works as a wrapper for a generic initialization routine.

Inheritance diagram for field_types_mod::generic_field_class:



Collaboration diagram for field_types_mod::generic_field_class:



Private Member Functions

- procedure test
- procedure inits1d
- inits2d
- inits3d
- inits4d
- procedure initv2d
- initv3d
- initv4d
- generic initialize => initS1D, initS2D, initS3D, initS4D, initV2D, initV3D, initV4D
- procedure print => printGenericField

Private Attributes

- type(scalar1d_field_class) scalar1d

    1D scalar field
- type(scalar2d_field_class) scalar2d

    2D scalar field
- type(scalar3d_field_class) scalar3d

    3D scalar field
- type(scalar4d_field_class) scalar4d

    4D scalar field
- type(vectorial2d_field_class) vectorial2d

    2D vectorial field
- type(vectorial3d_field_class) vectorial3d

    3D vectorial field
- type(vectorial4d_field_class) vectorial4d

    4D vectorial field

7.14.1 Detailed Description

generic field class. This works as a wrapper for a generic initialization routine.

Definition at line 104 of file fields_types.f90.

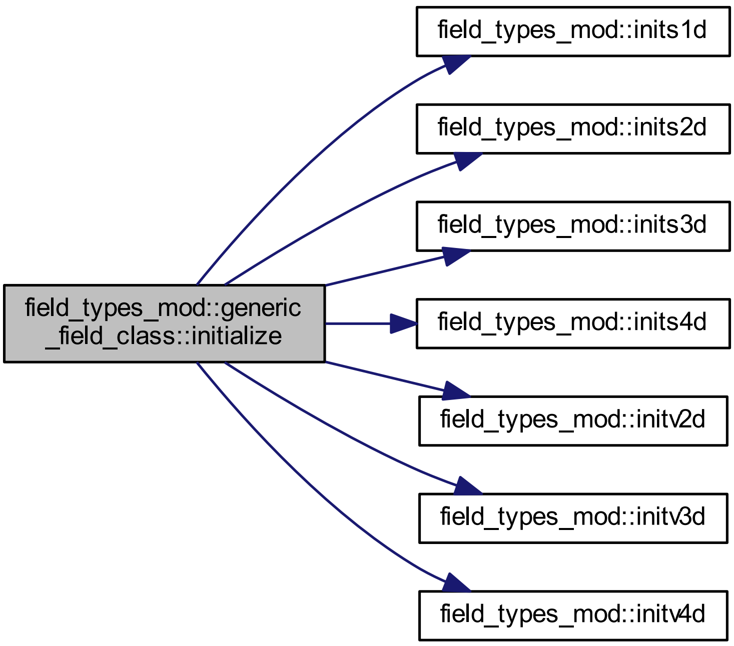
7.14.2 Member Function/Subroutine Documentation

7.14.2.1 initialize()

`generic field_types_mod::generic_field_class::initialize ( ) [private]`

Definition at line 116 of file fields_types.f90.

Here is the call graph for this function:



7.14.2.2 inits1d()

`procedure field_types_mod::generic_field_class::inits1d ( ) [private]`

Definition at line 114 of file fields_types.f90.

7.14.2.3 inits2d()

`field_types_mod::generic_field_class::inits2d ( ) [private]`

Definition at line 114 of file fields_types.f90.

7.14.2.4 inits3d()

`field_types_mod::generic_field_class::inits3d ( ) [private]`

Definition at line 114 of file fields_types.f90.

7.14.2.5 inits4d()

`field_types_mod::generic_field_class::inits4d ( ) [private]`

Definition at line 114 of file fields_types.f90.

7.14.2.6 initv2d()

`procedure field_types_mod::generic_field_class::initv2d ( ) [private]`

Definition at line 115 of file fields_types.f90.

7.14.2.7 initv3d()

`field_types_mod::generic_field_class::initv3d ( ) [private]`

Definition at line 115 of file fields_types.f90.

7.14.2.8 initv4d()

`field_types_mod::generic_field_class::initv4d ( ) [private]`

Definition at line 115 of file fields_types.f90.

7.14.2.9 print()

`procedure field_types_mod::generic_field_class::print ( ) [private]`

Definition at line 117 of file fields_types.f90.

7.14.2.10 test()

`procedure field_types_mod::generic_field_class::test ( ) [private]`

Definition at line 113 of file fields_types.f90.

7.14.3 Member Data Documentation

7.14.3.1 scalar1d

`type(scalar1d_field_class) field_types_mod::generic_field_class::scalar1d [private]`

1D scalar field

Definition at line 105 of file fields_types.f90.

```
105         type(scalar1d_field_class) :: scalar1d
```

7.14.3.2 scalar2d

`type(scalar2d_field_class) field_types_mod::generic_field_class::scalar2d [private]`

2D scalar field

Definition at line 106 of file fields_types.f90.

```
106         type(scalar2d_field_class) :: scalar2d
```

7.14.3.3 scalar3d

type(scalar3d_field_class) field_types_mod::generic_field_class::scalar3d [private]

3D scalar field

Definition at line 107 of file fields_types.f90.

```
107         type(scalar3d_field_class) :: scalar3d
```

7.14.3.4 scalar4d

type(scalar4d_field_class) field_types_mod::generic_field_class::scalar4d [private]

4D scalar field

Definition at line 108 of file fields_types.f90.

```
108         type(scalar4d_field_class) :: scalar4d
```

7.14.3.5 vectorial2d

type(vectorial2d_field_class) field_types_mod::generic_field_class::vectorial2d [private]

2D vectorial field

Definition at line 109 of file fields_types.f90.

```
109         type(vectorial2d_field_class) :: vectorial2d
```

7.14.3.6 vectorial3d

type(vectorial3d_field_class) field_types_mod::generic_field_class::vectorial3d [private]

3D vectorial field

Definition at line 110 of file fields_types.f90.

```
110         type(vectorial3d_field_class) :: vectorial3d
```

7.14.3.7 vectorial4d

`type(vectorial4d_field_class) field_types_mod::generic_field_class::vectorial4d [private]`

4D vectorial field

Definition at line 111 of file fields_types.f90.

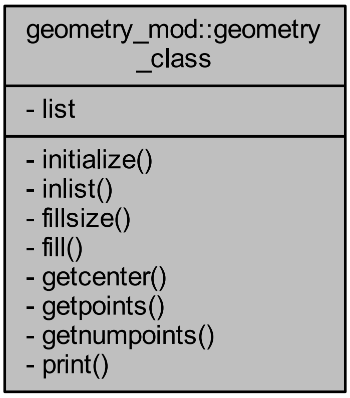
```
111         type(vectorial4d_field_class) :: vectorial4d
```

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

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

7.15 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 getcenter

  Function that retuns the shape baricenter.
- procedure getpoints

  Function that retuns the points (vertexes) that define the geometrical shape.
- procedure getnumpoints

  Function that returns the number of points (vertexes) that define the geometrical shape.
- procedure print => printGeometry

  prints the geometry type and contents

Private Attributes

- type(string), dimension(:), allocatable list
  String list with the name of possible geometry types.

7.15.1 Detailed Description

Definition at line 33 of file geometry.f90.

7.15.2 Member Function/Subroutine Documentation

7.15.2.1 fill()

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

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

Definition at line 39 of file geometry.f90.

7.15.2.2 fillsize()

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

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

Definition at line 38 of file geometry.f90.

7.15.2.3 getcenter()

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

Function that retuns the shape baricenter.

Definition at line 40 of file geometry.f90.

7.15.2.4 getnumpoints()

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

Function that returns the number of points (vertexes) that define the geometrical shape.

Definition at line 42 of file geometry.f90.

7.15.2.5 getpoints()

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

Function that retuns the points (vertexes) that define the geometrical shape.

Definition at line 41 of file geometry.f90.

7.15.2.6 initialize()

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

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

Definition at line 36 of file geometry.f90.

7.15.2.7 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 37 of file geometry.f90.

7.15.2.8 print()

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

prints the geometry type and contents

Definition at line 43 of file geometry.f90.

7.15.3 Member Data Documentation

7.15.3.1 list

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

String list with the name of possible geometry types.

Definition at line 34 of file geometry.f90.

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

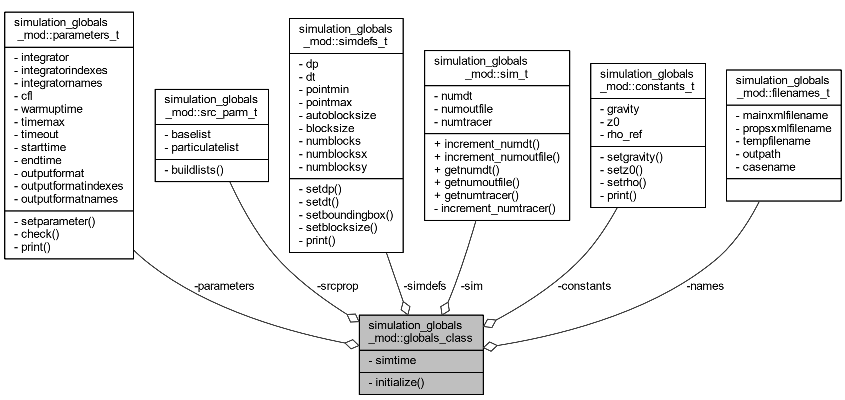
The documentation for this type was generated from the following file:

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

7.16 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) names
- real(prec_time) simtime
- type(src_parm_t) srcprop
- type(sim_t) sim

7.16.1 Detailed Description

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

Definition at line 110 of file simulation_globals.f90.

7.16.2 Member Function/Subroutine Documentation

7.16.2.1 initialize()

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

Definition at line 119 of file simulation_globals.f90.

7.16.3 Member Data Documentation

7.16.3.1 constants

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

Definition at line 113 of file simulation_globals.f90.

```
113         type(constants_t)    :: Constants
```

7.16.3.2 names

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

Definition at line 114 of file simulation_globals.f90.

```
114         type(filenames_t)    :: Names
```

7.16.3.3 parameters

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

Definition at line 111 of file simulation_globals.f90.

```
111         type(parameters_t)  :: Parameters
```

7.16.3.4 sim

`type(sim_t) simulation_globals_mod::globals_class::sim [private]`

Definition at line 117 of file simulation_globals.f90.

```
117         type(sim_t)         :: Sim
```

7.16.3.5 simdefs

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

Definition at line 112 of file simulation_globals.f90.

```
112         type(simdefs_t)     :: SimDefs
```

7.16.3.6 simtime

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

Definition at line 115 of file simulation_globals.f90.

```
115         real(prec_time)     :: SimTime
```

7.16.3.7 srcprop

```
type(src_parm_t) simulation_globals_mod::globals_class::srcprop [private]
```

Definition at line 116 of file simulation_globals.f90.

```
116         type(src_parm_t)    :: SrcProp
```

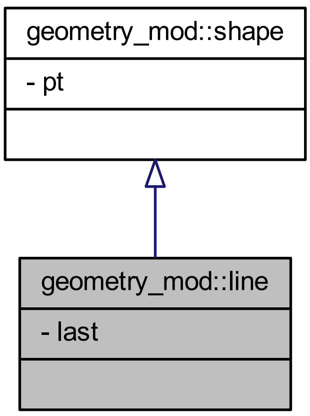
The documentation for this type was generated from the following file:

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

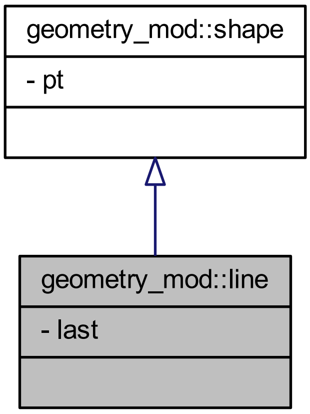
7.17 geometry_mod::line Type Reference

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.

7.17.1 Detailed Description

Type - line class.

Definition at line 53 of file geometry.f90.

7.17.2 Member Data Documentation

7.17.2.1 last

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

Coordinates of the end point.

Definition at line 54 of file geometry.f90.

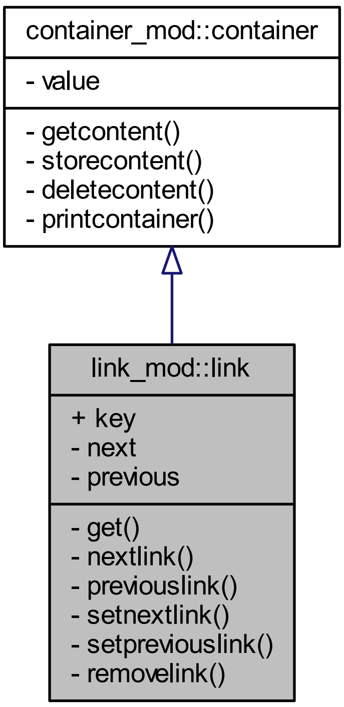
```
54         type(vector) :: last
```

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

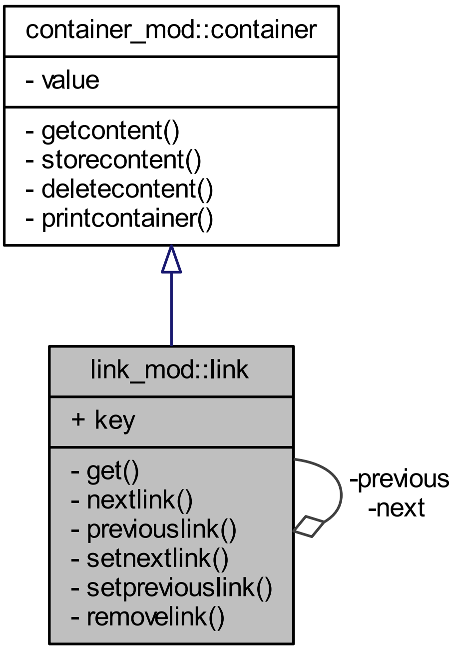
- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/geometry.f90

7.18 link_mod::link Interface Reference

Inheritance diagram for link_mod::link:



Collaboration diagram for link_mod::link:



Public Attributes

- integer, public key

Private Member Functions

- procedure get => getValue

    returns stored content
- procedure nextlink

    gets the next link
- procedure previouslink

    gets the previous link
- procedure setnextlink

    sets the next link pointer
- procedure setpreviouslink

    sets the previous link pointer
- procedure removelink

Private Attributes

- type(link), pointer next => null()

    pointer to a next link
- type(link), pointer previous => null()

    pointer to a previous link

7.18.1 Detailed Description

Definition at line 40 of file link.f90.

7.18.2 Member Function/Subroutine Documentation

7.18.2.1 get()

```
procedure link_mod::link::get ( )  [private]
```

returns stored content

Definition at line 46 of file link.f90.

7.18.2.2 nextlink()

```
procedure link_mod::link::nextlink ( )  [private]
```

gets the next link

Definition at line 47 of file link.f90.

7.18.2.3 previouslink()

```
procedure link_mod::link::previouslink ( )  [private]
```

gets the previous link

Definition at line 48 of file link.f90.

7.18.2.4 removelink()

```
procedure link_mod::link::removelink ( )  [private]
```

Definition at line 51 of file link.f90.

7.18.2.5 setnextlink()

```
procedure link_mod::link::setnextlink ( )  [private]
```

sets the next link pointer

Definition at line 49 of file link.f90.

7.18.2.6 setpreviouslink()

```
procedure link_mod::link::setpreviouslink ( )  [private]
```

sets the previous link pointer

Definition at line 50 of file link.f90.

7.18.3 Member Data Documentation

7.18.3.1 key

```
integer, public link_mod::link::key
```

Definition at line 42 of file link.f90.

```
42        integer, public :: key
```

7.18.3.2 next

```
type(link), pointer link_mod::link::next => null()  [private]
```

pointer to a next link

Definition at line 43 of file link.f90.

```
43        type(link), pointer :: next => null()
```

7.18.3.3 previous

```
type(link), pointer link_mod::link::previous => null()  [private]
```

pointer to a previous link

Definition at line 44 of file link.f90.

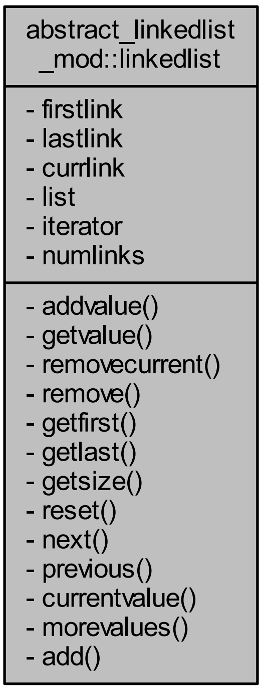
```
44        type(link), pointer :: previous => null()
```

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

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

7.19 abstract_linkedlist_mod::linkedlist Type Reference

Collaboration diagram for abstract_linkedlist_mod::linkedlist:



Private Member Functions

- procedure, non_overridable addvalue

    stores a value on the list
- procedure, non_overridable getvalue

    get nth value in list
- procedure, non_overridable removecurrent

    Method that removes the current link from a list.
- procedure, non_overridable remove

    Method that removes the nth link from a list.
- procedure, non_overridable getfirst

    returns the fist link of the list
- procedure, non_overridable getlast

    returns the last link of the list
- procedure, non_overridable getsize

    returns the size of the list
- procedure, non_overridable reset

    reset list iterator
- procedure, non_overridable next

    iterate to next value in list
- procedure, non_overridable previous

    iterate to previous value in list
- procedure, non_overridable currentvalue

    get current value in list
- procedure, non_overridable morevalues

    more values to iterate?
- generic add => addValue

Private Attributes

- class(link), pointer firstlink => null()

    First link in List.
- class(link), pointer lastlink => null()

    Last link in List.
- class(link), pointer currlink => null()
- class(link), pointer list
- class(link), pointer iterator
- integer numlinks = 0

7.19.1 Detailed Description

Definition at line 44 of file abstract_LinkedList.f90.

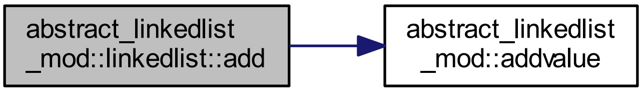
7.19.2 Member Function/Subroutine Documentation

7.19.2.1 add()

`generic abstract_linkedlist_mod::linkedlist::add ( ) [private]`

Definition at line 63 of file abstract_LinkedList.f90.

Here is the call graph for this function:



7.19.2.2 addvalue()

`procedure, non_overridable abstract_linkedlist_mod::linkedlist::addvalue ( ) [private]`

stores a value on the list

Definition at line 51 of file abstract_LinkedList.f90.

7.19.2.3 currentvalue()

`procedure, non_overridable abstract_linkedlist_mod::linkedlist::currentvalue ( ) [private]`

get current value in list

Definition at line 61 of file abstract_LinkedList.f90.

7.19.2.4 getfirst()

`procedure, non_overridable abstract_linkedlist_mod::linkedlist::getfirst ( ) [private]`

returns the fist link of the list

Definition at line 55 of file abstract_LinkedList.f90.

7.19.2.5 getlast()

`procedure, non_overridable abstract_linkedlist_mod::linkedlist::getlast ( )  [private]`

returns the last link of the list

Definition at line 56 of file abstract_LinkedList.f90.

7.19.2.6 getsize()

`procedure, non_overridable abstract_linkedlist_mod::linkedlist::getsize ( )  [private]`

returns the size of the list

Definition at line 57 of file abstract_LinkedList.f90.

7.19.2.7 getvalue()

`procedure, non_overridable abstract_linkedlist_mod::linkedlist::getvalue ( )  [private]`

get nth value in list

Definition at line 52 of file abstract_LinkedList.f90.

7.19.2.8 morevalues()

`procedure, non_overridable abstract_linkedlist_mod::linkedlist::morevalues ( )  [private]`

more values to iterate?

Definition at line 62 of file abstract_LinkedList.f90.

7.19.2.9 next()

`procedure, non_overridable abstract_linkedlist_mod::linkedlist::next ( )  [private]`

iterate to next value in list

Definition at line 59 of file abstract_LinkedList.f90.

7.19.2.10 previous()

procedure, non_overridable abstract_linkedlist_mod::linkedlist::previous ( ) [private]

iterate to previous value in list

Definition at line 60 of file abstract_LinkedList.f90.

7.19.2.11 remove()

procedure, non_overridable abstract_linkedlist_mod::linkedlist::remove ( ) [private]

Method that removes the nth link from a list.

Definition at line 54 of file abstract_LinkedList.f90.

7.19.2.12 removecurrent()

procedure, non_overridable abstract_linkedlist_mod::linkedlist::removecurrent ( ) [private]

Method that removes the current link from a list.

Definition at line 53 of file abstract_LinkedList.f90.

7.19.2.13 reset()

procedure, non_overridable abstract_linkedlist_mod::linkedlist::reset ( ) [private]

reset list iterator

Definition at line 58 of file abstract_LinkedList.f90.

7.19.3 Member Data Documentation

7.19.3.1 currlink

class(link), pointer abstract_linkedlist_mod::linkedlist::currlink => null() [private]

Definition at line 48 of file abstract_LinkedList.f90.

```
48         class(link), pointer :: currLink => null()
```

7.19.3.2 firstlink

```
class(link), pointer abstract_linkedlist_mod::linkedlist::firstlink => null() [private]
```

First link in List.

Definition at line 46 of file abstract_LinkedList.f90.

```
46         class(link), pointer :: firstLink => null()
```

7.19.3.3 iterator

```
class(link), pointer abstract_linkedlist_mod::linkedlist::iterator [private]
```

Definition at line 48 of file abstract_LinkedList.f90.

7.19.3.4 lastlink

```
class(link), pointer abstract_linkedlist_mod::linkedlist::lastlink => null() [private]
```

Last link in List.

Definition at line 47 of file abstract_LinkedList.f90.

```
47         class(link), pointer :: lastLink => null()
```

7.19.3.5 list

```
class(link), pointer abstract_linkedlist_mod::linkedlist::list [private]
```

Definition at line 48 of file abstract_LinkedList.f90.

7.19.3.6 numlinks

```
integer abstract_linkedlist_mod::linkedlist::numlinks = 0 [private]
```

Definition at line 49 of file abstract_LinkedList.f90.

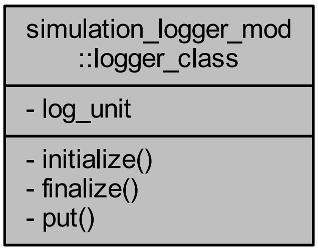
```
49         integer :: numLinks = 0
```

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

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

7.20 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

7.20.1 Detailed Description

Definition at line 29 of file simulation_logger.f90.

7.20.2 Member Function/Subroutine Documentation

7.20.2.1 finalize()

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

Definition at line 34 of file simulation_logger.f90.

7.20.2.2 initialize()

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

Definition at line 33 of file simulation_logger.f90.

7.20.2.3 put()

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

Definition at line 35 of file simulation_logger.f90.

7.20.3 Member Data Documentation

7.20.3.1 log_unit

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

Definition at line 31 of file simulation_logger.f90.

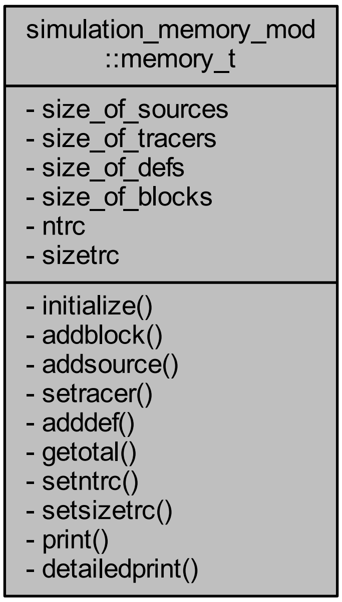
```
31         integer :: log_unit = -1
```

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

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

7.21 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 setracer
- procedure adddef
- procedure getotal
- procedure setntrc
- procedure setsizetrc
- 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)

- integer ntrc

  Expected number of Tracers for the simulation (by Source emission at least)

- integer sizetrc

  Size of a dummy Tracer, in bytes.

7.21.1 Detailed Description

Definition at line 28 of file simulation_memory.f90.

7.21.2 Member Function/Subroutine Documentation

7.21.2.1 addblock()

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

Definition at line 38 of file simulation_memory.f90.

7.21.2.2 adddef()

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

Definition at line 41 of file simulation_memory.f90.

7.21.2.3 addsource()

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

Definition at line 39 of file simulation_memory.f90.

7.21.2.4 detailedprint()

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

Definition at line 46 of file simulation_memory.f90.

7.21.2.5 getotal()

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

Definition at line 42 of file simulation_memory.f90.

7.21.2.6 initialize()

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

Definition at line 37 of file simulation_memory.f90.

7.21.2.7 print()

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

Definition at line 45 of file simulation_memory.f90.

7.21.2.8 setntrc()

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

Definition at line 43 of file simulation_memory.f90.

7.21.2.9 setracer()

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

Definition at line 40 of file simulation_memory.f90.

7.21.2.10 setsizetrc()

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

Definition at line 44 of file simulation_memory.f90.

7.21.3 Member Data Documentation

7.21.3.1 ntrc

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

Expected number of Tracers for the simulation (by Source emission at least)

Definition at line 34 of file simulation_memory.f90.

```
34         integer :: ntrc
```

7.21.3.2 size_of_blocks

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

Size of the Blocks in memory (bytes)

Definition at line 33 of file simulation_memory.f90.

```
33         integer :: size_of_blocks
```

7.21.3.3 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 32 of file simulation_memory.f90.

```
32         integer :: size_of_defs
```

7.21.3.4 size_of_sources

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

Size of the sources in memory (bytes)

Definition at line 30 of file simulation_memory.f90.

```
30         integer :: size_of_sources
```

7.21.3.5 size_of_tracers

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

Size of the tracers in memory (bytes)

Definition at line 31 of file simulation_memory.f90.

```
31         integer :: size_of_tracers
```

7.21.3.6 sizetrc

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

Size of a dummy Tracer, in bytes.

Definition at line 35 of file simulation_memory.f90.

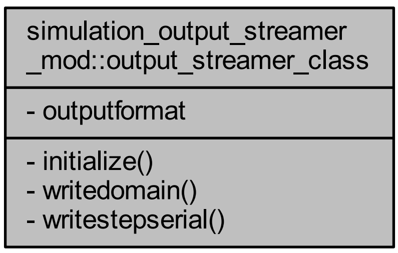
```
35         integer :: sizeTrc
```

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

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

7.22 simulation_output_streamer_mod::output_streamer_class Type Reference

Collaboration diagram for simulation_output_streamer_mod::output_streamer_class:



Private Member Functions

- procedure initialize => initOutputStreamer
- procedure writedomain
- procedure writestepserial

Private Attributes

- integer outputformat = -1

7.22.1 Detailed Description

Definition at line 31 of file simulation_output_streamer.f90.

7.22.2 Member Function/Subroutine Documentation

7.22.2.1 initialize()

```
procedure simulation_output_streamer_mod::output_streamer_class::initialize ( ) [private]
```

Definition at line 34 of file simulation_output_streamer.f90.

7.22.2.2 writedomain()

```
procedure simulation_output_streamer_mod::output_streamer_class::writedomain ( ) [private]
```

Definition at line 35 of file simulation_output_streamer.f90.

7.22.2.3 writestepserial()

```
procedure simulation_output_streamer_mod::output_streamer_class::writestepserial ( ) [private]
```

Definition at line 36 of file simulation_output_streamer.f90.

7.22.3 Member Data Documentation

7.22.3.1 outputformat

```
integer simulation_output_streamer_mod::output_streamer_class::outputformat = -1 [private]
```

Definition at line 32 of file simulation_output_streamer.f90.

```
32        integer :: OutputFormat = -1
```

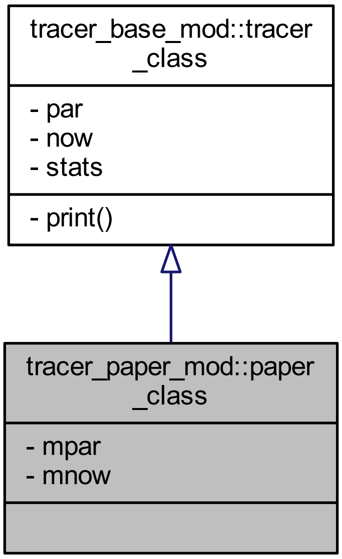
The documentation for this type was generated from the following file:

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

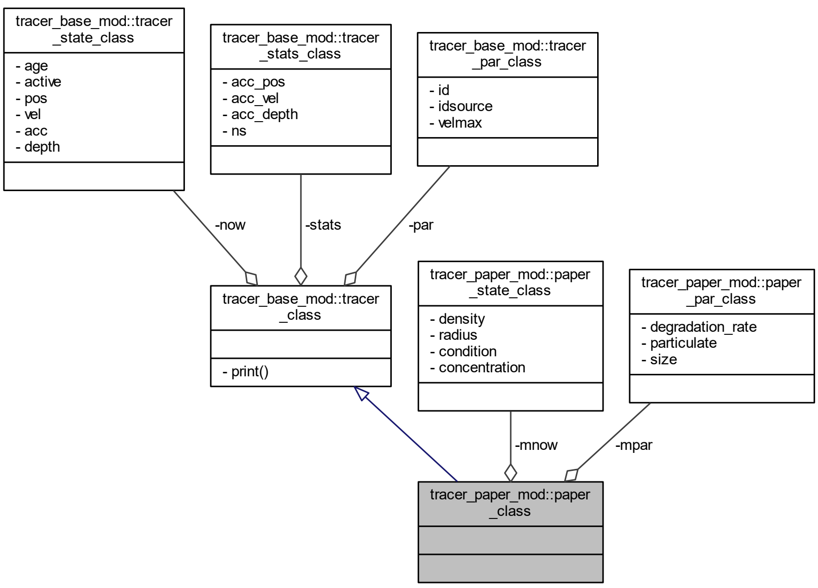
7.23 tracer_paper_mod::paper_class Type Reference

Type - The plastic material Lagrangian tracer class.

Inheritance diagram for tracer_paper_mod::paper_class:



Collaboration diagram for tracer_paper_mod::paper_class:



Private Attributes

- type(paper_par_class) mpar
  To access material parameters.
- type(paper_state_class) mnow
  To access material state variables.

7.23.1 Detailed Description

Type - The plastic material Lagrangian tracer class.

Definition at line 43 of file tracer_paper.f90.

7.23.2 Member Data Documentation

7.23.2.1 mnow

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

To access material state variables.

Definition at line 45 of file tracer_paper.f90.

```
45        type(paper_state_class) :: mnow
```

7.23.2.2 mpar

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

To access material parameters.

Definition at line 44 of file tracer_paper.f90.

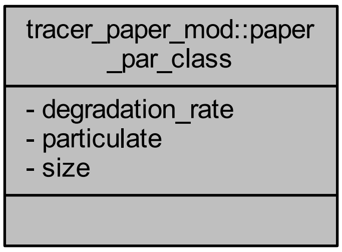
```
44        type(paper_par_class)   :: mpar
```

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

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

7.24 tracer_paper_mod::paper_par_class Type Reference

Collaboration diagram for tracer_paper_mod::paper_par_class:



Private Attributes

- 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)

7.24.1 Detailed Description

Definition at line 30 of file tracer_paper.f90.

7.24.2 Member Data Documentation

7.24.2.1 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
```

7.24.2.2 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
```

7.24.2.3 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
```

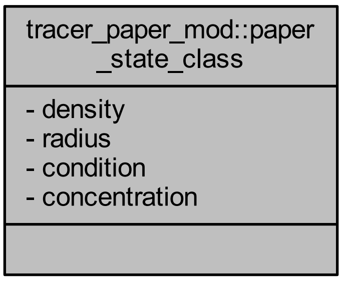
The documentation for this type was generated from the following file:

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

7.25 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) density

  density of the material
- real(prec) radius

  Tracer radius (m)
- real(prec) condition

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

  Particle concentration.

7.25.1 Detailed Description

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

Definition at line 36 of file tracer_paper.f90.

7.25.2 Member Data Documentation

7.25.2.1 concentration

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

Particle concentration.

Definition at line 40 of file tracer_paper.f90.

```
40        real(prec) :: concentration
```

7.25.2.2 condition

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

Material condition (1-0)

Definition at line 39 of file tracer_paper.f90.

```
39        real(prec) :: condition
```

7.25.2.3 density

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

density of the material

Definition at line 37 of file tracer_paper.f90.

```
37         real(prec) :: density
```

7.25.2.4 radius

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

Tracer radius (m)

Definition at line 38 of file tracer_paper.f90.

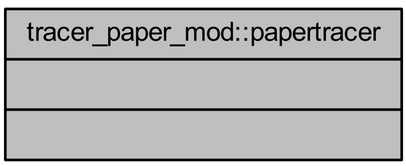
```
38         real(prec) :: radius
```

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

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

7.26 tracer_paper_mod::papertracer Interface Reference

Collaboration diagram for tracer_paper_mod::papertracer:



7.26.1 Detailed Description

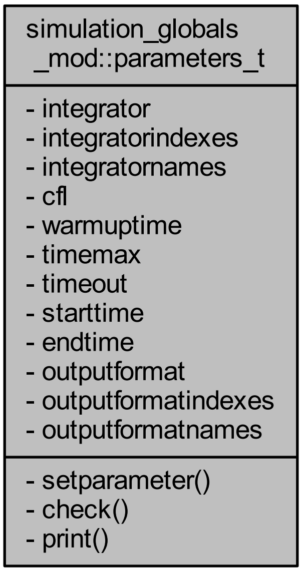
Definition at line 56 of file tracer_paper.f90.

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

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

7.27 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)
- integer, dimension(3) integratorindexes

  Index list for the integrator selector.
- type(string), dimension(3) integratornames

  Names list for the integrator selector.
- real(prec) cfl = 0.5

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

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

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

  Time out data (1/Hz)

- type(datetime) starttime

  Start date of the simulation.
- type(datetime) endtime

  End date of the simulation.
- integer outputformat = 2

  Format of the output files (default=2) NetCDF=1, VTK=2.
- integer, dimension(2) outputformatindexes

  Index list for the output file format selector.
- type(string), dimension(2) outputformatnames

  Names list for the output file format selector.

7.27.1 Detailed Description

Definition at line 34 of file simulation_globals.f90.

7.27.2 Member Function/Subroutine Documentation

7.27.2.1 check()

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

Definition at line 49 of file simulation_globals.f90.

7.27.2.2 print()

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

Definition at line 50 of file simulation_globals.f90.

7.27.2.3 setparameter()

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

Definition at line 48 of file simulation_globals.f90.

7.27.3 Member Data Documentation

7.27.3.1 cfl

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

Courant Friedrichs Lewy condition number.

Definition at line 38 of file simulation_globals.f90.

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

7.27.3.2 endtime

`type(datetime) simulation_globals_mod::parameters_t::endtime [private]`

End date of the simulation.

Definition at line 43 of file simulation_globals.f90.

```
43        type(datetime) :: EndTime
```

7.27.3.3 integrator

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

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

Definition at line 35 of file simulation_globals.f90.

```
35        integer    :: Integrator = 1
```

7.27.3.4 integratorindexes

`integer, dimension(3) simulation_globals_mod::parameters_t::integratorindexes [private]`

Index list for the integrator selector.

Definition at line 36 of file simulation_globals.f90.

```
36        integer    :: IntegratorIndexes(3)
```

7.27.3.5 integratornames

`type(string), dimension(3) simulation_globals_mod::parameters_t::integratornames [private]`

Names list for the integrator selector.

Definition at line 37 of file simulation_globals.f90.

```
37        type(string) :: IntegratorNames(3)
```

7.27.3.6 outputformat

`integer simulation_globals_mod::parameters_t::outputformat = 2 [private]`

Format of the output files (default=2) NetCDF=1, VTK=2.

Definition at line 44 of file simulation_globals.f90.

```
44        integer    :: OutputFormat = 2
```

7.27.3.7 outputformatindexes

`integer, dimension(2) simulation_globals_mod::parameters_t::outputformatindexes [private]`

Index list for the output file format selector.

Definition at line 45 of file simulation_globals.f90.

```
45        integer    :: OutputFormatIndexes(2)
```

7.27.3.8 outputformatnames

`type(string), dimension(2) simulation_globals_mod::parameters_t::outputformatnames [private]`

Names list for the output file format selector.

Definition at line 46 of file simulation_globals.f90.

```
46        type(string) :: OutputFormatNames(2)
```

7.27.3.9 starttime

```
type(datetime) simulation_globals_mod::parameters_t::starttime [private]
```

Start date of the simulation.

Definition at line 42 of file simulation_globals.f90.

```
42        type(datetime) :: StartTime
```

7.27.3.10 timemax

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

Simulation duration (s)

Definition at line 40 of file simulation_globals.f90.

```
40        real(prec_time) :: TimeMax = mv
```

7.27.3.11 timeout

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

Time out data (1/Hz)

Definition at line 41 of file simulation_globals.f90.

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

7.27.3.12 warmuptime

```
real(prec_time) 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 39 of file simulation_globals.f90.

```
39        real(prec_time) :: WarmUpTime = 0.0
```

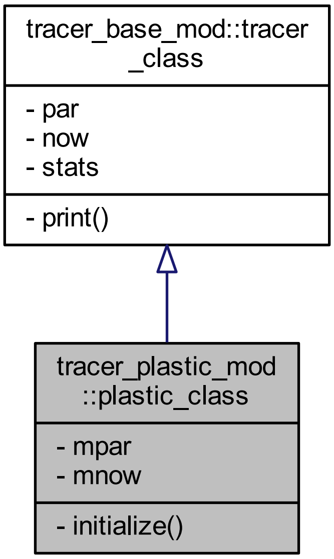
The documentation for this type was generated from the following file:

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

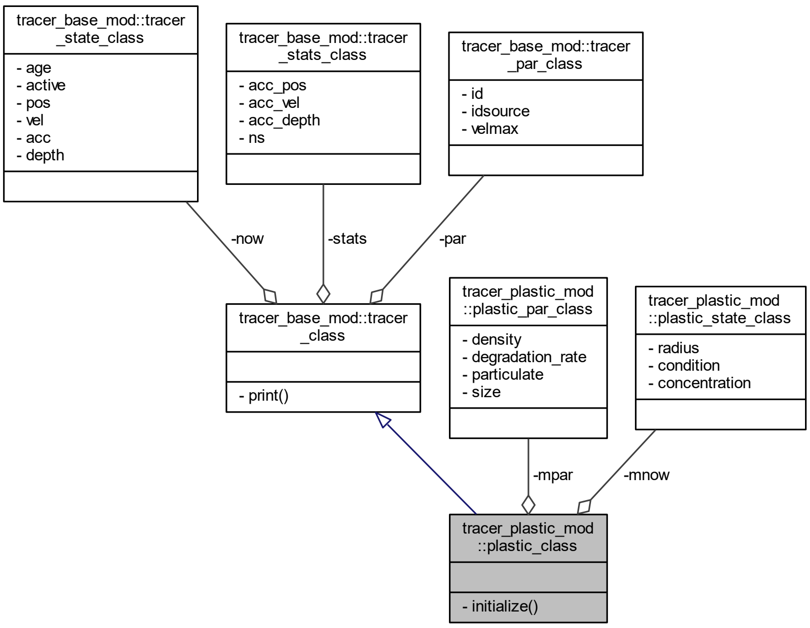
7.28 tracer_plastic_mod::plastic_class Type Reference

Type - The plastic material Lagrangian tracer class.

Inheritance diagram for tracer_plastic_mod::plastic_class:



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.

7.28.1 Detailed Description

Type - The plastic material Lagrangian tracer class.

Definition at line 42 of file tracer_plastic.f90.

7.28.2 Member Function/Subroutine Documentation

7.28.2.1 initialize()

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

Definition at line 46 of file tracer_plastic.f90.

7.28.3 Member Data Documentation

7.28.3.1 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
```

7.28.3.2 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.

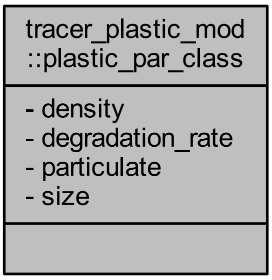
```
43        type(plastic_par_class)   :: mpar
```

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

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

7.29 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)

7.29.1 Detailed Description

Definition at line 29 of file tracer_plastic.f90.

7.29.2 Member Data Documentation

7.29.2.1 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
```

7.29.2.2 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
```

7.29.2.3 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
```

7.29.2.4 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
```

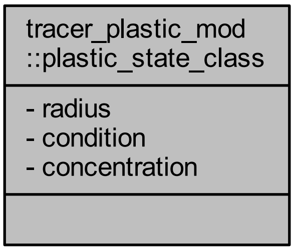
The documentation for this type was generated from the following file:

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

7.30 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.

7.30.1 Detailed Description

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

Definition at line 36 of file tracer_plastic.f90.

7.30.2 Member Data Documentation

7.30.2.1 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
```

7.30.2.2 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
```

7.30.2.3 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
```

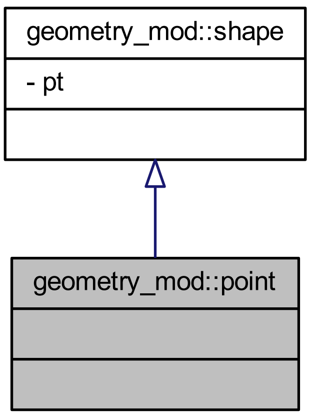
The documentation for this type was generated from the following file:

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

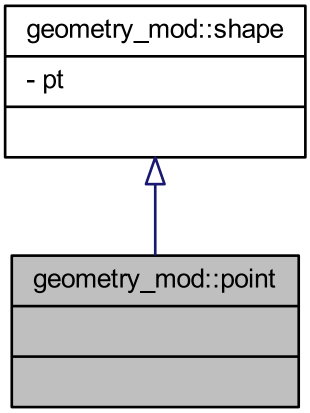
7.31 geometry_mod::point Type Reference

Type - point class.

Inheritance diagram for geometry_mod::point:



Collaboration diagram for geometry_mod::point:



7.31.1 Detailed Description

Type - point class.

Definition at line 50 of file geometry.f90.

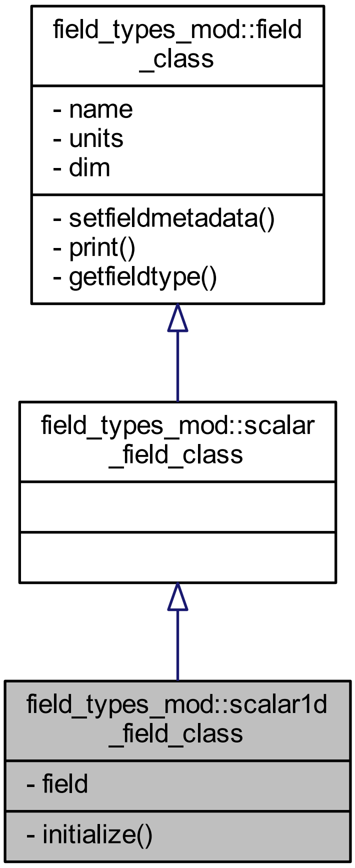
The documentation for this type was generated from the following file:

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

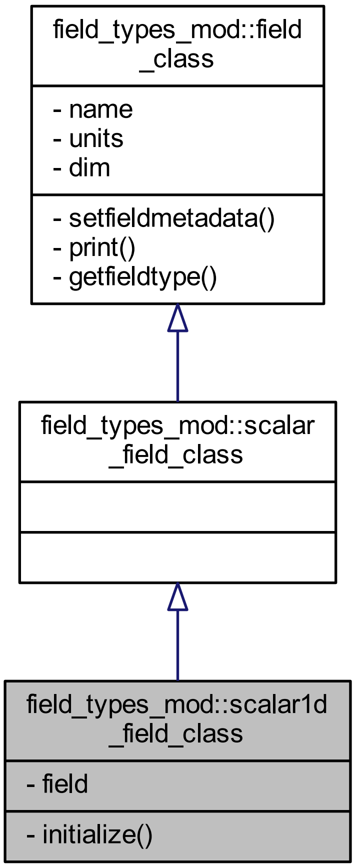
7.32 field_types_mod::scalar1d_field_class Type Reference

a 1D scalar field class

Inheritance diagram for field_types_mod::scalar1d_field_class:



Collaboration diagram for field_types_mod::scalar1d_field_class:



Private Member Functions

- procedure initialize => initScalar1dField

Private Attributes

- real(prec), dimension(:), allocatable field

    the data on the scalar data field

7.32.1 Detailed Description

a 1D scalar field class

Definition at line 51 of file fields_types.f90.

7.32.2 Member Function/Subroutine Documentation

7.32.2.1 initialize()

```
procedure field_types_mod::scalar1d_field_class::initialize ( )  [private]
```

Definition at line 54 of file fields_types.f90.

7.32.3 Member Data Documentation

7.32.3.1 field

```
real(prec), dimension(:), allocatable field_types_mod::scalar1d_field_class::field  [private]
```

the data on the scalar data field

Definition at line 52 of file fields_types.f90.

```
52        real(prec), allocatable, dimension(:) :: field
```

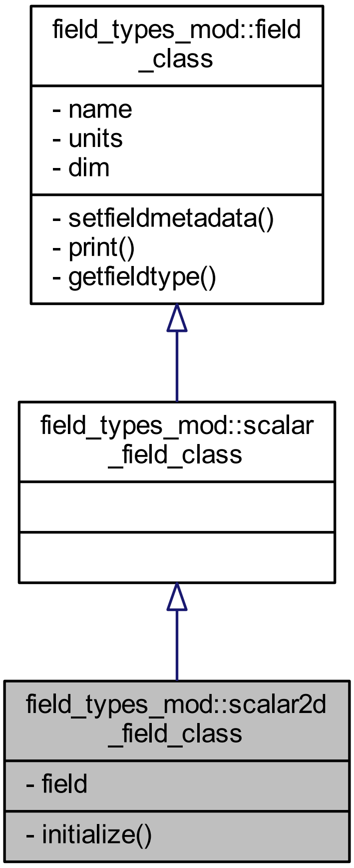
The documentation for this type was generated from the following file:

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

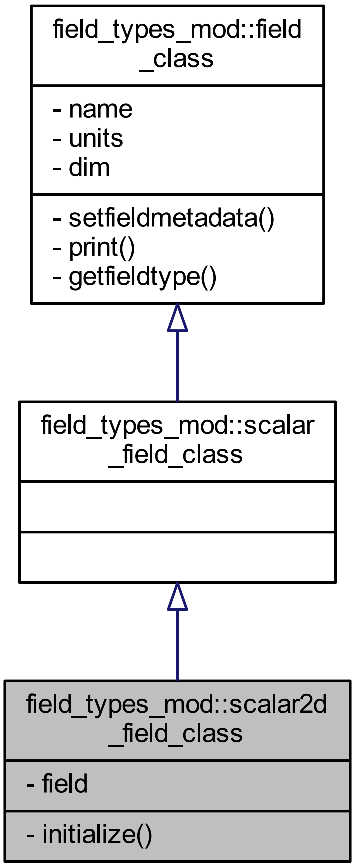
7.33 field_types_mod::scalar2d_field_class Type Reference

a 2D scalar field class

Inheritance diagram for field_types_mod::scalar2d_field_class:



Collaboration diagram for field_types_mod::scalar2d_field_class:



Private Member Functions

- procedure initialize => initScalar2dField

Private Attributes

- real(prec), dimension(:,:), allocatable field

  the data on the scalar data field

7.33.1 Detailed Description

a 2D scalar field class

Definition at line 57 of file fields_types.f90.

7.33.2 Member Function/Subroutine Documentation

7.33.2.1 initialize()

```
procedure field_types_mod::scalar2d_field_class::initialize ( )  [private]
```

Definition at line 60 of file fields_types.f90.

7.33.3 Member Data Documentation

7.33.3.1 field

```
real(prec), dimension(:,:), allocatable field_types_mod::scalar2d_field_class::field  [private]
```

the data on the scalar data field

Definition at line 58 of file fields_types.f90.

```
58        real(prec), allocatable, dimension(:,:) :: field
```

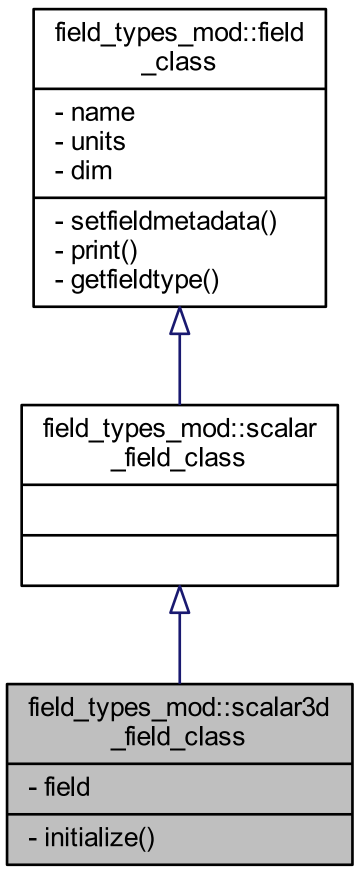
The documentation for this type was generated from the following file:

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

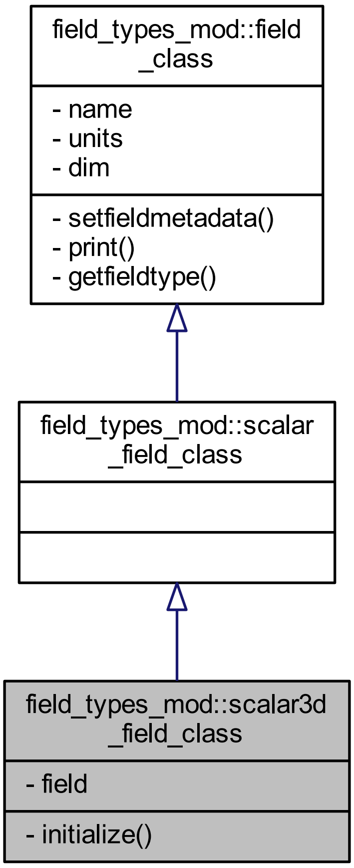
7.34 field_types_mod::scalar3d_field_class Type Reference

a 3D scalar field class

Inheritance diagram for field_types_mod::scalar3d_field_class:



Collaboration diagram for field_types_mod::scalar3d_field_class:



Private Member Functions

- procedure initialize => initScalar3dField

Private Attributes

- real(prec), dimension(:,:,:), allocatable field

    the data on the scalar data field

7.34.1 Detailed Description

a 3D scalar field class

Definition at line 63 of file fields_types.f90.

7.34.2 Member Function/Subroutine Documentation

7.34.2.1 initialize()

```
procedure field_types_mod::scalar3d_field_class::initialize ( )  [private]
```

Definition at line 66 of file fields_types.f90.

7.34.3 Member Data Documentation

7.34.3.1 field

```
real(prec), dimension(:,:,:), allocatable field_types_mod::scalar3d_field_class::field  [private]
```

the data on the scalar data field

Definition at line 64 of file fields_types.f90.

```
64         real(prec), allocatable, dimension(:,:,:) :: field
```

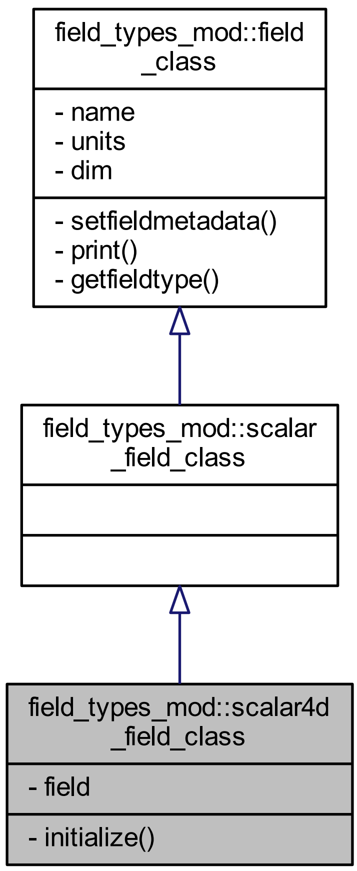
The documentation for this type was generated from the following file:

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

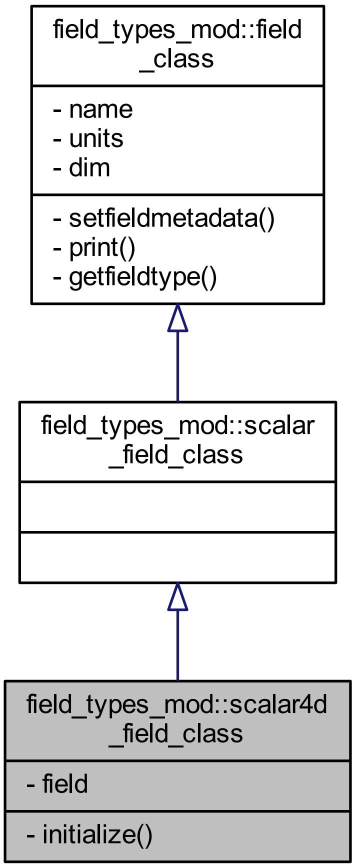
7.35 field_types_mod::scalar4d_field_class Type Reference

a 4D scalar field class

Inheritance diagram for field_types_mod::scalar4d_field_class:



Collaboration diagram for field_types_mod::scalar4d_field_class:



Private Member Functions

- procedure initialize => initScalar4dField

Private Attributes

- real(prec), dimension(:,:,:,:), allocatable field

  the data on the scalar data field

7.35.1 Detailed Description

a 4D scalar field class

Definition at line 69 of file fields_types.f90.

7.35.2 Member Function/Subroutine Documentation

7.35.2.1 initialize()

```
procedure field_types_mod::scalar4d_field_class::initialize ( )  [private]
```

Definition at line 72 of file fields_types.f90.

7.35.3 Member Data Documentation

7.35.3.1 field

```
real(prec), dimension(:,:,:,:), allocatable field_types_mod::scalar4d_field_class::field  [private]
```

the data on the scalar data field

Definition at line 70 of file fields_types.f90.

```
70        real(prec), allocatable, dimension(:,:,:,:) :: field
```

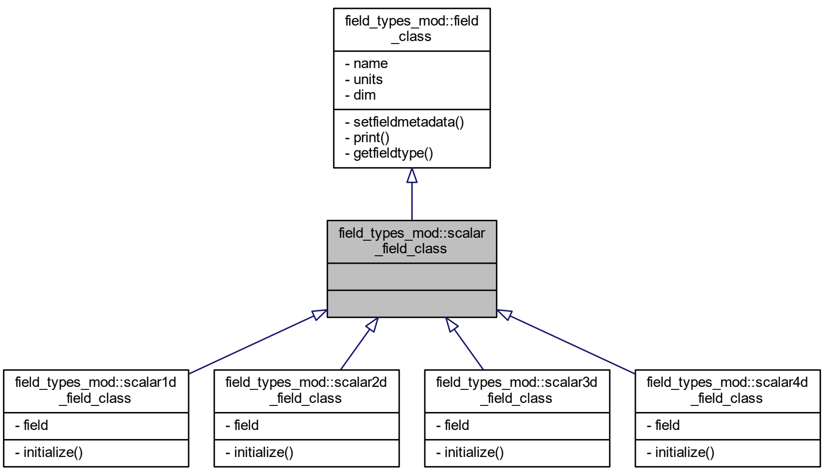
The documentation for this type was generated from the following file:

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

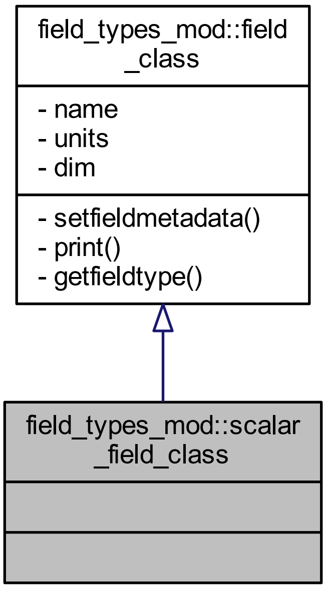
7.36 field_types_mod::scalar_field_class Type Reference

a scalar field class

Inheritance diagram for field_types_mod::scalar_field_class:



Collaboration diagram for field_types_mod::scalar_field_class:



7.36.1 Detailed Description

a scalar field class

Definition at line 47 of file fields_types.f90.

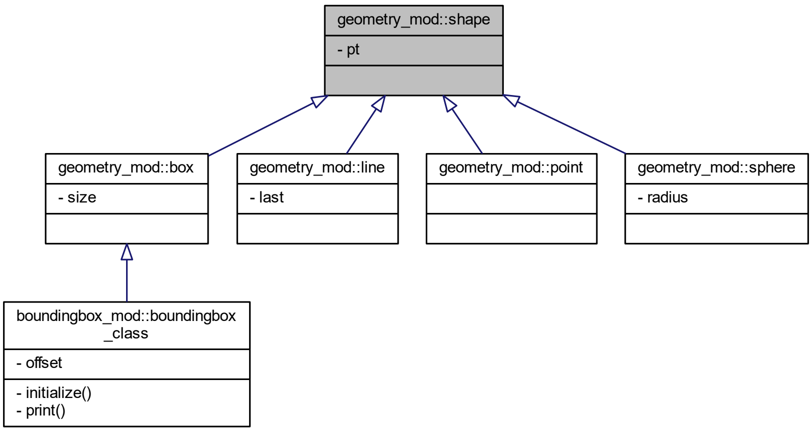
The documentation for this type was generated from the following file:

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

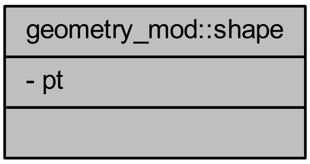
7.37 geometry_mod::shape Type Reference

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.

7.37.1 Detailed Description

Type - extendable shape class.

Definition at line 46 of file geometry.f90.

7.37.2 Member Data Documentation

7.37.2.1 pt

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

Coordinates of a point.

Definition at line 47 of file geometry.f90.

```
47         type(vector) :: pt
```

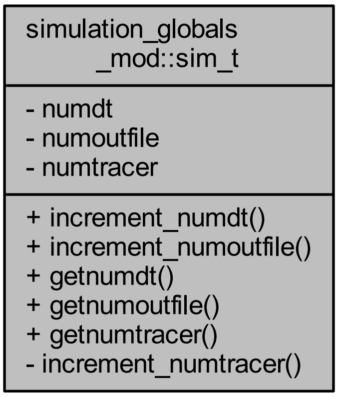
The documentation for this type was generated from the following file:

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

7.38 simulation_globals_mod::sim_t Type Reference

Simulation related counters and others.

Collaboration diagram for simulation_globals_mod::sim_t:



Public Member Functions

- procedure, public increment_numdt
- procedure, public increment_numoutfile
- procedure, public getnumdt
- procedure, public getnumoutfile
- procedure, public getnumtracer

Private Member Functions

- procedure, private increment_numtracer

Private Attributes

- integer numdt

  number of the current iteration
- integer numoutfile

  number of the current output file
- integer numtracer

  Global Tracer number holder. Incremented at tracer construction or first activation time.

7.38.1 Detailed Description

Simulation related counters and others.

Definition at line 96 of file simulation_globals.f90.

7.38.2 Member Function/Subroutine Documentation

7.38.2.1 getnumdt()

```
procedure, public simulation_globals_mod::sim_t::getnumdt ( )
```

Definition at line 104 of file simulation_globals.f90.

7.38.2.2 getnumoutfile()

```
procedure, public simulation_globals_mod::sim_t::getnumoutfile ( )
```

Definition at line 105 of file simulation_globals.f90.

7.38.2.3 getnumtracer()

```
procedure, public simulation_globals_mod::sim_t::getnumtracer ( )
```

Definition at line 107 of file simulation_globals.f90.

7.38.2.4 increment_numdt()

```
procedure, public simulation_globals_mod::sim_t::increment_numdt ( )
```

Definition at line 102 of file simulation_globals.f90.

7.38.2.5 increment_numoutfile()

procedure, public simulation_globals_mod::sim_t::increment_numoutfile ( )

Definition at line 103 of file simulation_globals.f90.

7.38.2.6 increment_numtracer()

procedure, private simulation_globals_mod::sim_t::increment_numtracer ( ) [private]

Definition at line 106 of file simulation_globals.f90.

7.38.3 Member Data Documentation

7.38.3.1 numdt

integer simulation_globals_mod::sim_t::numdt [private]

number of the current iteration

Definition at line 98 of file simulation_globals.f90.

```
98        integer :: numdt
```

7.38.3.2 numoutfile

integer simulation_globals_mod::sim_t::numoutfile [private]

number of the current output file

Definition at line 99 of file simulation_globals.f90.

```
99        integer :: numoutfile
```

7.38.3.3 numtracer

```
integer simulation_globals_mod::sim_t::numtracer  [private]
```

Global Tracer number holder. Incremented at tracer construction or first activation time.

Definition at line 100 of file simulation_globals.f90.

```
100        integer :: numTracer
```

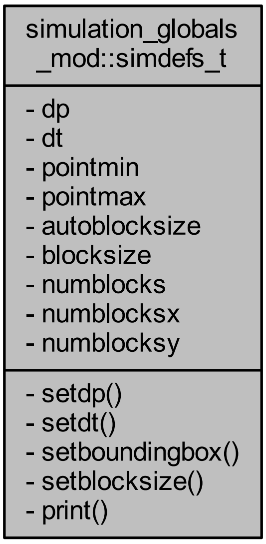
The documentation for this type was generated from the following file:

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

7.39 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

  Initial particle spacing at emission.
- 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 (xyz) of a Block (sub-domain)
- integer numblocks

  Number of blocks in the simulation.
- integer numblocksx
- integer numblocksy

  Number of blocks along x and y.

7.39.1 Detailed Description

Simulation definitions class.

Definition at line 53 of file simulation_globals.f90.

7.39.2 Member Function/Subroutine Documentation

7.39.2.1 print()

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

Definition at line 67 of file simulation_globals.f90.

7.39.2.2 setblocksize()

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

Definition at line 66 of file simulation_globals.f90.

7.39.2.3 setboundingbox()

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

Definition at line 65 of file simulation_globals.f90.

7.39.2.4 setdp()

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

Definition at line 63 of file simulation_globals.f90.

7.39.2.5 setdt()

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

Definition at line 64 of file simulation_globals.f90.

7.39.3 Member Data Documentation

7.39.3.1 autoblocksize

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

Flag for automatic Block sizing.

Definition at line 58 of file simulation_globals.f90.

```
58        logical          ::  autoblocksize = .true.
```

7.39.3.2 blocksize

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

Size (xyz) of a Block (sub-domain)

Definition at line 59 of file simulation_globals.f90.

```
59        type(vector)     ::  blocksize
```

7.39.3.3 dp

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

Initial particle spacing at emission.

Definition at line 54 of file simulation_globals.f90.

```
54         real(prec)      ::  Dp
```

7.39.3.4 dt

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

Timestep for fixed step integrators (s)

Definition at line 55 of file simulation_globals.f90.

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

7.39.3.5 numblocks

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

Number of blocks in the simulation.

Definition at line 60 of file simulation_globals.f90.

```
60         integer         ::  numblocks
```

7.39.3.6 numblocksx

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

Definition at line 61 of file simulation_globals.f90.

```
61         integer         ::  numblocksx, numblocksy
```

7.39.3.7 numblocksy

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

Number of blocks along x and y.

Definition at line 61 of file simulation_globals.f90.

7.39.3.8 pointmax

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

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

Definition at line 57 of file simulation_globals.f90.

```
57        type(vector)    ::  Pointmax
```

7.39.3.9 pointmin

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

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

Definition at line 56 of file simulation_globals.f90.

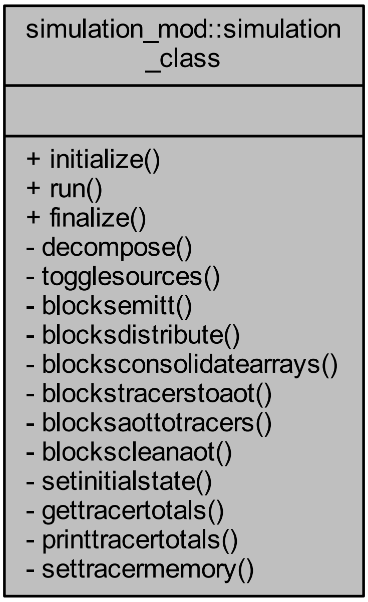
```
56        type(vector)    ::  Pointmin
```

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

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

7.40 simulation_mod::simulation_class Type Reference

Collaboration diagram for simulation_mod::simulation_class:



Public Member Functions

- procedure, public initialize => initSimulation
- procedure, public run
- procedure, public finalize => closeSimulation

Private Member Functions

- procedure, private decompose => DecomposeDomain
- procedure, private togglesources
- procedure, private blocksemitt
- procedure, private blocksdistribute
- procedure, private blocksconsolidatearrays
- procedure, private blockstracerstoaot
- procedure, private blocksaottotracers
- procedure, private blockscleanaot
- procedure, private setinitialstate
- procedure, private gettracertotals
- procedure, private printtracertotals
- procedure, private settracermemory

7.40.1 Detailed Description

Definition at line 38 of file simulation.f90.

7.40.2 Member Function/Subroutine Documentation

7.40.2.1 blocksaottotracers()

```
procedure, private simulation_mod::simulation_class::blocksaottotracers ( )  [private]
```

Definition at line 49 of file simulation.f90.

7.40.2.2 blockscleanaot()

```
procedure, private simulation_mod::simulation_class::blockscleanaot ( )  [private]
```

Definition at line 50 of file simulation.f90.

7.40.2.3 blocksconsolidatearrays()

```
procedure, private simulation_mod::simulation_class::blocksconsolidatearrays ( )  [private]
```

Definition at line 47 of file simulation.f90.

7.40.2.4 blocksdistribute()

```
procedure, private simulation_mod::simulation_class::blocksdistribute ( )  [private]
```

Definition at line 46 of file simulation.f90.

7.40.2.5 blocksemitt()

```
procedure, private simulation_mod::simulation_class::blocksemitt ( )  [private]
```

Definition at line 45 of file simulation.f90.

7.40.2.6 blockstracerstoaot()

procedure, private simulation_mod::simulation_class::blockstracerstoaot ( ) [private]

Definition at line 48 of file simulation.f90.

7.40.2.7 decompose()

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

Definition at line 43 of file simulation.f90.

7.40.2.8 finalize()

procedure, public simulation_mod::simulation_class::finalize ( )

Definition at line 42 of file simulation.f90.

7.40.2.9 gettracertotals()

procedure, private simulation_mod::simulation_class::gettracertotals ( ) [private]

Definition at line 52 of file simulation.f90.

7.40.2.10 initialize()

procedure, public simulation_mod::simulation_class::initialize ( )

Definition at line 40 of file simulation.f90.

7.40.2.11 printtracertotals()

procedure, private simulation_mod::simulation_class::printtracertotals ( ) [private]

Definition at line 53 of file simulation.f90.

7.40.2.12 run()

```
procedure, public simulation_mod::simulation_class::run ( )
```

Definition at line 41 of file simulation.f90.

7.40.2.13 setinitialstate()

```
procedure, private simulation_mod::simulation_class::setinitialstate ( ) [private]
```

Definition at line 51 of file simulation.f90.

7.40.2.14 settracermemory()

```
procedure, private simulation_mod::simulation_class::settracermemory ( ) [private]
```

Definition at line 54 of file simulation.f90.

7.40.2.15 togglesources()

```
procedure, private simulation_mod::simulation_class::togglesources ( ) [private]
```

Definition at line 44 of file simulation.f90.

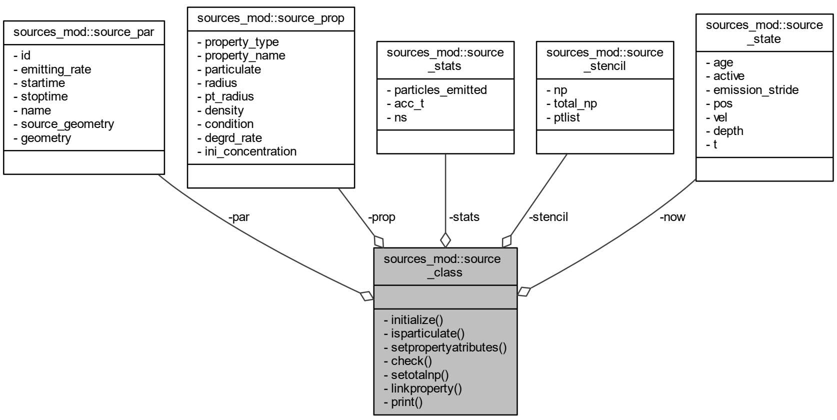
The documentation for this type was generated from the following file:

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

7.41 sources_mod::source_class Type Reference

Type - The source class.

Collaboration diagram for sources_mod::source_class:



Private Member Functions

- procedure initialize => initializeSource
- procedure isparticulate
- procedure setpropertyatributes
- procedure check
- procedure, private setotalnp
- procedure, private linkproperty
- procedure print => printSource

Private Attributes

- type(source_par) par

  To access parameters.
- type(source_prop) prop

  To access Tracer properties.
- type(source_state) now

  To access state variables.
- type(source_stencil) stencil

  To acess stencil variables.
- type(source_stats) stats

  To access statistics.

7.41.1 Detailed Description

Type - The source class.

Definition at line 73 of file sources.f90.

7.41.2 Member Function/Subroutine Documentation

7.41.2.1 check()

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

Definition at line 83 of file sources.f90.

7.41.2.2 initialize()

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

Definition at line 80 of file sources.f90.

7.41.2.3 isparticulate()

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

Definition at line 81 of file sources.f90.

7.41.2.4 linkproperty()

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

Definition at line 85 of file sources.f90.

7.41.2.5 print()

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

Definition at line 86 of file sources.f90.

7.41.2.6 setotalnp()

```
procedure, private sources_mod::source_class::setotalnp ( )  [private]
```

Definition at line 84 of file sources.f90.

7.41.2.7 setpropertyatributes()

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

Definition at line 82 of file sources.f90.

7.41.3 Member Data Documentation

7.41.3.1 now

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

To access state variables.

Definition at line 76 of file sources.f90.

```
76         type(source_state) :: now
```

7.41.3.2 par

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

To access parameters.

Definition at line 74 of file sources.f90.

```
74         type(source_par)   :: par
```

7.41.3.3 prop

`type(source_prop) sources_mod::source_class::prop [private]`

To access Tracer properties.

Definition at line 75 of file sources.f90.

```
75         type(source_prop)  :: prop
```

7.41.3.4 stats

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

To access statistics.

Definition at line 78 of file sources.f90.

```
78         type(source_stats) :: stats
```

7.41.3.5 stencil

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

To acess stencil variables.

Definition at line 77 of file sources.f90.

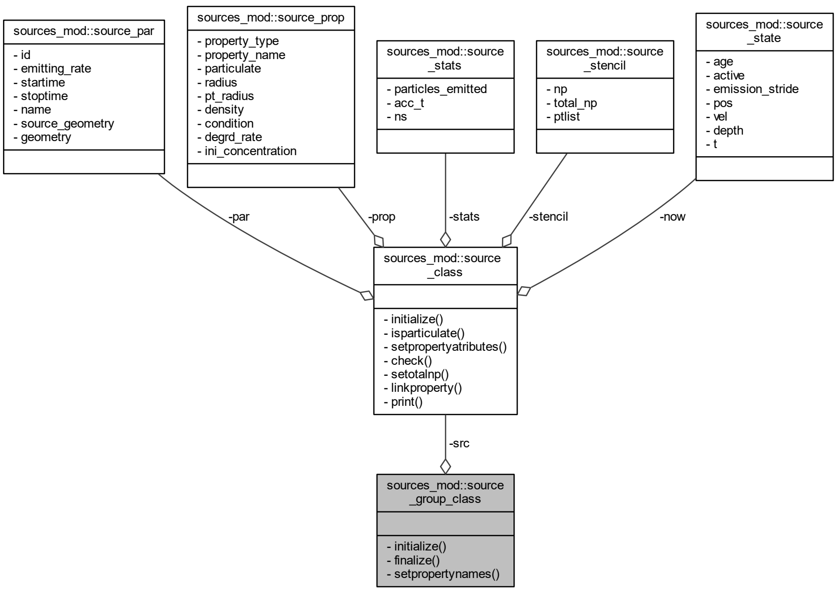
```
77         type(source_stencil) :: stencil
```

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

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

7.42 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 setpropertynames

Private Attributes

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

7.42.1 Detailed Description

Definition at line 89 of file sources.f90.

7.42.2 Member Function/Subroutine Documentation

7.42.2.1 finalize()

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

Definition at line 93 of file sources.f90.

7.42.2.2 initialize()

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

Definition at line 92 of file sources.f90.

7.42.2.3 setpropertynames()

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

Definition at line 94 of file sources.f90.

7.42.3 Member Data Documentation

7.42.3.1 src

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

Definition at line 90 of file sources.f90.

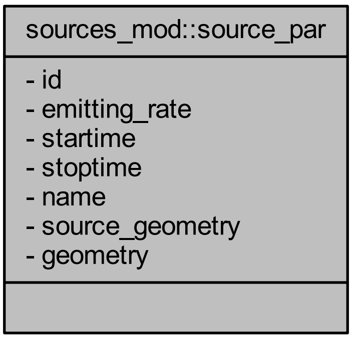
```
90         type(source_class), allocatable, dimension(:) :: src
```

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

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

7.43 sources_mod::source_par Type Reference

Collaboration diagram for sources_mod::source_par:



Private Attributes

- integer id
  unique source identification (integer)
- integer 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) source_geometry
  Source type : 'point', 'line', 'sphere', 'box'.
- class(shape), allocatable geometry
  Source geometry.

7.43.1 Detailed Description

Definition at line 27 of file sources.f90.

7.43.2 Member Data Documentation

7.43.2.1 emitting_rate

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

Emitting rate of the source (Hz)

Definition at line 29 of file sources.f90.

```
29          integer :: emitting_rate
```

7.43.2.2 geometry

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

Source geometry.

Definition at line 34 of file sources.f90.

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

7.43.2.3 id

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

unique source identification (integer)

Definition at line 28 of file sources.f90.

```
28          integer :: id
```

7.43.2.4 name

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

source name

Definition at line 32 of file sources.f90.

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

7.43.2.5 source_geometry

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

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

Definition at line 33 of file sources.f90.

```
33         type(string) :: source_geometry
```

7.43.2.6 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
```

7.43.2.7 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
```

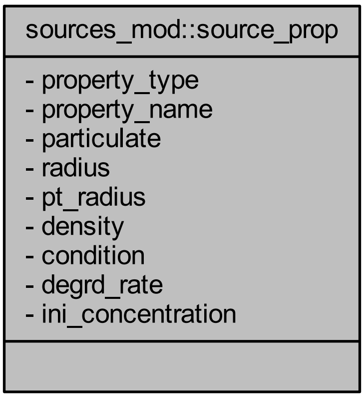
The documentation for this type was generated from the following file:

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

7.44 sources_mod::source_prop Type Reference

Type - material properties of a source object.

Collaboration diagram for sources_mod::source_prop:



Private Attributes

- type(string) property_type

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

  source property name
- logical particulate

  true for a Source that emitts particulate tracers (a concentration of particles)
- real(prec) radius

  radius of the emitted Tracers (size of the particle if not particulate, volume of the Tracer if particulate)
- real(prec) pt_radius

  radius of the emitted particles (Tracers if not particulate)
- real(prec) density

  density of the Tracers
- real(prec) condition

  condition of the Tracers
- real(prec) degrd_rate

  degradation rate of the Tracers
- real(prec) ini_concentration

  initial concentration of particles if particulate

7.44.1 Detailed Description

Type - material properties of a source object.

Definition at line 37 of file sources.f90.

7.44.2 Member Data Documentation

7.44.2.1 condition

`real(prec) sources_mod::source_prop::condition [private]`

condition of the Tracers

Definition at line 44 of file sources.f90.

```
44 real(prec) :: condition
```

7.44.2.2 degrd_rate

`real(prec) sources_mod::source_prop::degrd_rate [private]`

degradation rate of the Tracers

Definition at line 45 of file sources.f90.

```
45 real(prec) :: degrd_rate
```

7.44.2.3 density

`real(prec) sources_mod::source_prop::density [private]`

density of the Tracers

Definition at line 43 of file sources.f90.

```
43 real(prec) :: density
```

7.44.2.4 ini_concentration

`real(prec) sources_mod::source_prop::ini_concentration [private]`

initial concentration of particles if particulate

Definition at line 46 of file sources.f90.

```
46 real(prec) :: ini_concentration
```

7.44.2.5 particulate

`logical sources_mod::source_prop::particulate [private]`

true for a Source that emitts particulate tracers (a concentration of particles)

Definition at line 40 of file sources.f90.

```
40         logical :: particulate
```

7.44.2.6 property_name

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

source property name

Definition at line 39 of file sources.f90.

```
39         type(string) :: property_name
```

7.44.2.7 property_type

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

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

Definition at line 38 of file sources.f90.

```
38         type(string) :: property_type
```

7.44.2.8 pt_radius

`real(prec) sources_mod::source_prop::pt_radius [private]`

radius of the emitted particles (Tracers if not particulate)

Definition at line 42 of file sources.f90.

```
42         real(prec) :: pt_radius
```

7.44.2.9 radius

```
real(prec) sources_mod::source_prop::radius [private]
```

radius of the emitted Tracers (size of the particle if not particulate, volume of the Tracer if particulate)

Definition at line 41 of file sources.f90.

```
41        real(prec) :: radius
```

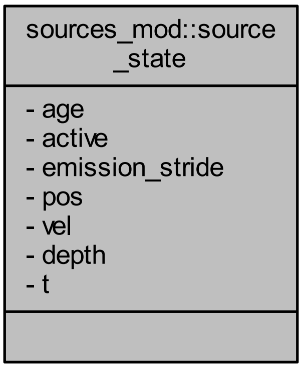
The documentation for this type was generated from the following file:

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

7.45 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
- integer emission_stride
  Number of time steps to wait until next emission.
- 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)

7.45.1 Detailed Description

Type - state variables of a source object.

Definition at line 49 of file sources.f90.

7.45.2 Member Data Documentation

7.45.2.1 active

`logical sources_mod::source_state::active [private]`

active switch

Definition at line 51 of file sources.f90.

```
51         logical :: active
```

7.45.2.2 age

`real(prec_time) sources_mod::source_state::age [private]`

Definition at line 50 of file sources.f90.

```
50         real(prec_time) :: age              ! time variables
```

7.45.2.3 depth

`real(prec) sources_mod::source_state::depth [private]`

Depth of the source baricenter (m)

Definition at line 55 of file sources.f90.

```
55         real(prec) :: depth
```

7.45.2.4 emission_stride

```
integer sources_mod::source_state::emission_stride [private]
```

Number of time steps to wait until next emission.

Definition at line 52 of file sources.f90.

```
52     integer :: emission_stride
```

7.45.2.5 pos

```
type(vector) sources_mod::source_state::pos [private]
```

Position of the source baricenter (m)

Definition at line 53 of file sources.f90.

```
53     type(vector) :: pos
```

7.45.2.6 t

```
real(prec) sources_mod::source_state::t [private]
```

Temperature of the source (Celcius)

Definition at line 56 of file sources.f90.

```
56     real(prec) :: T
```

7.45.2.7 vel

```
type(vector) sources_mod::source_state::vel [private]
```

Velocity of the source (m s-1)

Definition at line 54 of file sources.f90.

```
54     type(vector) :: vel
```

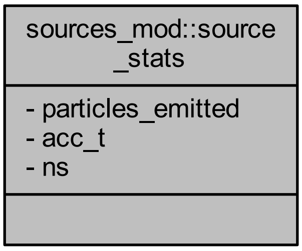
The documentation for this type was generated from the following file:

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

7.46 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.

7.46.1 Detailed Description

Type - statistical variables of a source object.

Definition at line 59 of file sources.f90.

7.46.2 Member Data Documentation

7.46.2.1 acc_t

```
real(prec_wrt) sources_mod::source_stats::acc_t [private]
```

Accumulated temperature of the tracer (Celcius)

Definition at line 63 of file sources.f90.

```
63        real(prec_wrt) :: acc_T
```

7.46.2.2 ns

`integer sources_mod::source_stats::ns [private]`

Number of sampling steps.

Definition at line 64 of file sources.f90.

```
64         integer :: ns
```

7.46.2.3 particles_emitted

`integer sources_mod::source_stats::particles_emitted [private]`

Number of emitted particles by this source.

Definition at line 62 of file sources.f90.

```
62         integer :: particles_emitted
```

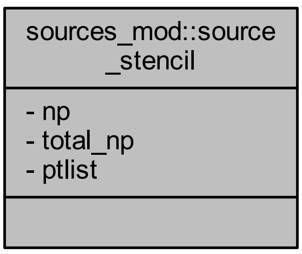
The documentation for this type was generated from the following file:

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

7.47 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.

7.47.1 Detailed Description

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

Definition at line 67 of file sources.f90.

7.47.2 Member Data Documentation

7.47.2.1 np

```
integer sources_mod::source_stencil::np [private]
```

Number of tracers by emission.

Definition at line 68 of file sources.f90.

```
68         integer :: np
```

7.47.2.2 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 70 of file sources.f90.

```
70         type(vector), allocatable, dimension(:) :: ptlist
```

7.47.2.3 total_np

```
integer sources_mod::source_stencil::total_np  [private]
```

Total number of tracers that this source will generate.

Definition at line 69 of file sources.f90.

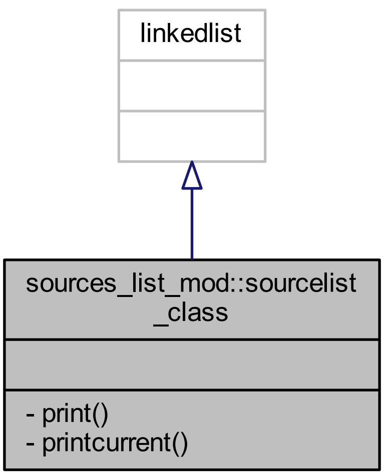
```
69        integer :: total_np
```

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

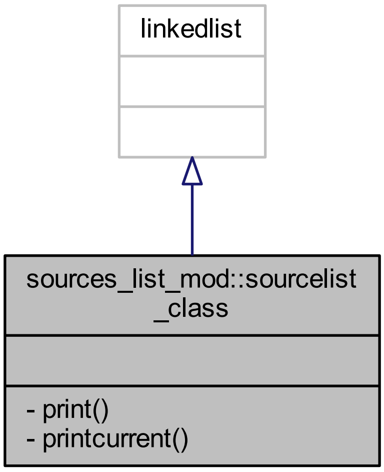
- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/sources.f90

7.48 sources_list_mod::sourcelist_class Type Reference

Inheritance diagram for sources_list_mod::sourcelist_class:



Collaboration diagram for sources_list_mod::sourcelist_class:



Private Member Functions

- procedure print => print_sourceList
- procedure printcurrent => print_sourceListCurrent

7.48.1 Detailed Description

Definition at line 31 of file sources_list.f90.

7.48.2 Member Function/Subroutine Documentation

7.48.2.1 print()

```
procedure sources_list_mod::sourcelist_class::print ( ) [private]
```

Definition at line 33 of file sources_list.f90.

7.48.2.2 printcurrent()

```
procedure sources_list_mod::sourcelist_class::printcurrent ( ) [private]
```

Definition at line 34 of file sources_list.f90.

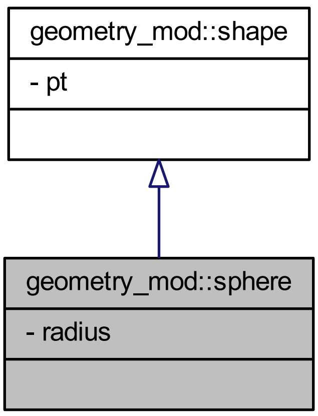
The documentation for this type was generated from the following file:

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

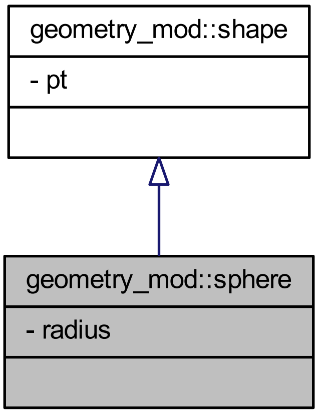
7.49 geometry_mod::sphere Type Reference

Type - sphere class.

Inheritance diagram for geometry_mod::sphere:



Collaboration diagram for geometry_mod::sphere:



Private Attributes

- real(prec) radius
  Sphere radius (m)

7.49.1 Detailed Description

Type - sphere class.

Definition at line 57 of file geometry.f90.

7.49.2 Member Data Documentation

7.49.2.1 radius

```
real(prec) geometry_mod::sphere::radius  [private]
```

Sphere radius (m)

Definition at line 58 of file geometry.f90.

```
58          real(prec) :: radius
```

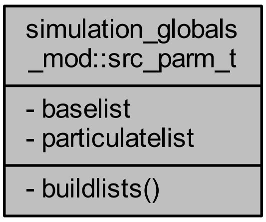
The documentation for this type was generated from the following file:

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

7.50 simulation_globals_mod::src_parm_t Type Reference

Lists for Source parameters.

Collaboration diagram for simulation_globals_mod::src_parm_t:



Private Member Functions

- procedure buildlists

Private Attributes

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

    Lists for base tracer parameters.
- type(string), dimension(:), allocatable particulatelist

    List for parameters of particulate type tracers.

7.50.1 Detailed Description

Lists for Source parameters.

Definition at line 89 of file simulation_globals.f90.

7.50.2 Member Function/Subroutine Documentation

7.50.2.1 buildlists()

```
procedure simulation_globals_mod::src_parm_t::buildlists ( )  [private]
```

Definition at line 93 of file simulation_globals.f90.

7.50.3 Member Data Documentation

7.50.3.1 baselist

`type(string), dimension(:), allocatable simulation_globals_mod::src_parm_t::baselist [private]`

Lists for base tracer parameters.

Definition at line 90 of file simulation_globals.f90.

```
90         type(string), allocatable, dimension(:) :: baselist
```

7.50.3.2 particulatelist

`type(string), dimension(:), allocatable simulation_globals_mod::src_parm_t::particulatelist [private]`

List for parameters of particulate type tracers.

Definition at line 91 of file simulation_globals.f90.

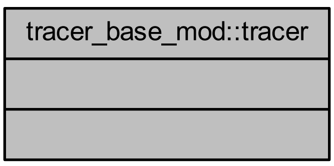
```
91         type(string), allocatable, dimension(:) :: particulatelist
```

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

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

7.51 tracer_base_mod::tracer Interface Reference

Collaboration diagram for tracer_base_mod::tracer:



7.51.1 Detailed Description

Definition at line 70 of file tracer_base.f90.

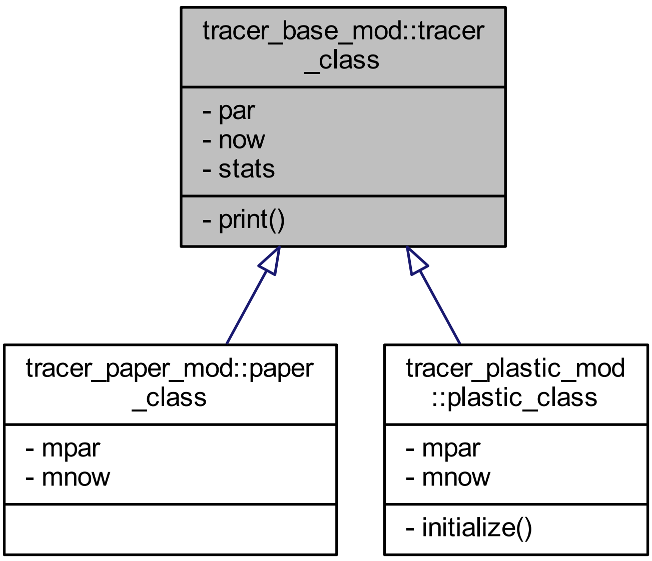
The documentation for this interface was generated from the following file:

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

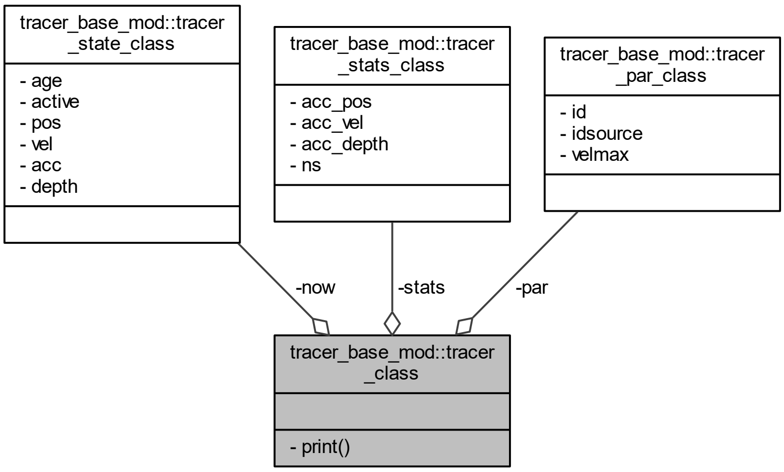
7.52 tracer_base_mod::tracer_class Type Reference

Type - The pure Lagrangian tracer class.

Inheritance diagram for tracer_base_mod::tracer_class:



Collaboration diagram for tracer_base_mod::tracer_class:



Private Member Functions

- procedure print => printTracer

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.

7.52.1 Detailed Description

Type - The pure Lagrangian tracer class.

Definition at line 53 of file tracer_base.f90.

7.52.2 Member Function/Subroutine Documentation

7.52.2.1 print()

```
procedure tracer_base_mod::tracer_class::print ( )  [private]
```

Definition at line 58 of file tracer_base.f90.

7.52.3 Member Data Documentation

7.52.3.1 now

type(tracer_state_class) tracer_base_mod::tracer_class::now [private]

To access state variables.

Definition at line 55 of file tracer_base.f90.

```
55         type(tracer_state_class) :: now
```

7.52.3.2 par

type(tracer_par_class) tracer_base_mod::tracer_class::par [private]

To access parameters.

Definition at line 54 of file tracer_base.f90.

```
54         type(tracer_par_class)   :: par
```

7.52.3.3 stats

type(tracer_stats_class) tracer_base_mod::tracer_class::stats [private]

To access statistics.

Definition at line 56 of file tracer_base.f90.

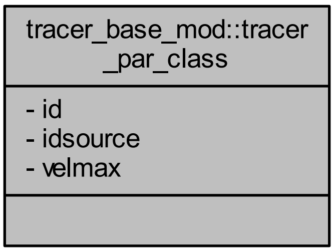
```
56         type(tracer_stats_class) :: stats
```

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

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

7.53 tracer_base_mod::tracer_par_class Type Reference

Collaboration diagram for tracer_base_mod::tracer_par_class:



Private Attributes

- integer id = MV
  unique tracer identification
- integer idsource = MV
  Source to which the tracer belongs.
- real(prec) velmax = MV
  Maximum velocity of tracer to track (m/s)

7.53.1 Detailed Description

Definition at line 27 of file tracer_base.f90.

7.53.2 Member Data Documentation

7.53.2.1 id

`integer tracer_base_mod::tracer_par_class::id = MV [private]`

unique tracer identification

Definition at line 28 of file tracer_base.f90.

```
28        integer :: id = mv
```

7.53.2.2 idsource

```
integer tracer_base_mod::tracer_par_class::idsource = MV  [private]
```

Source to which the tracer belongs.

Definition at line 29 of file tracer_base.f90.

```
29         integer :: idsource = mv
```

7.53.2.3 velmax

```
real(prec) tracer_base_mod::tracer_par_class::velmax = MV  [private]
```

Maximum velocity of tracer to track (m/s)

Definition at line 30 of file tracer_base.f90.

```
30         real(prec) :: velmax = mv
```

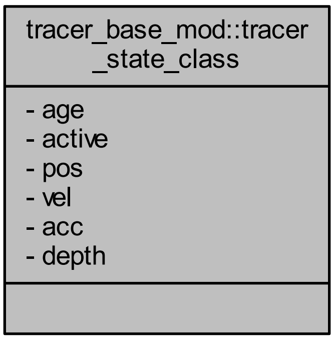
The documentation for this type was generated from the following file:

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

7.54 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 = MV
- logical active = .false.

    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 = MV

    Depth of the tracer (m)

7.54.1 Detailed Description

Type - state variables of a pure Lagrangian tracer object.

Definition at line 33 of file tracer_base.f90.

7.54.2 Member Data Documentation

7.54.2.1 acc

```
type(vector) tracer_base_mod::tracer_state_class::acc [private]
```

Acceleration of the tracer (m s-2)

Definition at line 38 of file tracer_base.f90.

```
38          type(vector) :: acc
```

7.54.2.2 active

```
logical tracer_base_mod::tracer_state_class::active = .false. [private]
```

active switch

Definition at line 35 of file tracer_base.f90.

```
35          logical :: active = .false.
```

7.54.2.3 age

```
real(prec_time) tracer_base_mod::tracer_state_class::age = MV  [private]
```

Definition at line 34 of file tracer_base.f90.

```
34          real(prec_time) :: age = mv              ! time variables
```

7.54.2.4 depth

```
real(prec) tracer_base_mod::tracer_state_class::depth = MV  [private]
```

Depth of the tracer (m)

Definition at line 39 of file tracer_base.f90.

```
39          real(prec) :: depth = mv
```

7.54.2.5 pos

```
type(vector) tracer_base_mod::tracer_state_class::pos  [private]
```

Position of the tracer (m)

Definition at line 36 of file tracer_base.f90.

```
36          type(vector) :: pos
```

7.54.2.6 vel

```
type(vector) tracer_base_mod::tracer_state_class::vel  [private]
```

Velocity of the tracer (m s-1)

Definition at line 37 of file tracer_base.f90.

```
37          type(vector) :: vel
```

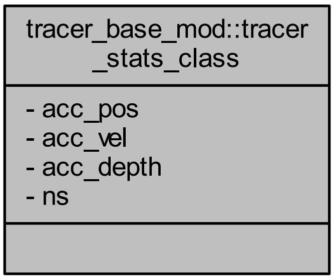
The documentation for this type was generated from the following file:

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

7.55 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 = MV

  Accumulated depth of the tracer (m)
- integer ns = MV

  Number of sampling steps.

7.55.1 Detailed Description

Type - statistical variables of a pure Lagrangian tracer object.

Definition at line 43 of file tracer_base.f90.

7.55.2 Member Data Documentation

7.55.2.1 acc_depth

```
real(prec_wrt) tracer_base_mod::tracer_stats_class::acc_depth = MV  [private]
```

Accumulated depth of the tracer (m)

Definition at line 48 of file tracer_base.f90.

```
48         real(prec_wrt) :: acc_depth = mv
```

7.55.2.2 acc_pos

```
type(vector) tracer_base_mod::tracer_stats_class::acc_pos [private]
```

Accumulated position of the tracer (m)

Definition at line 46 of file tracer_base.f90.

```
46         type(vector) :: acc_pos
```

7.55.2.3 acc_vel

```
type(vector) tracer_base_mod::tracer_stats_class::acc_vel [private]
```

Accumulated velocity of the tracer (m s-1)

Definition at line 47 of file tracer_base.f90.

```
47         type(vector) :: acc_vel
```

7.55.2.4 ns

```
integer tracer_base_mod::tracer_stats_class::ns = MV [private]
```

Number of sampling steps.

Definition at line 50 of file tracer_base.f90.

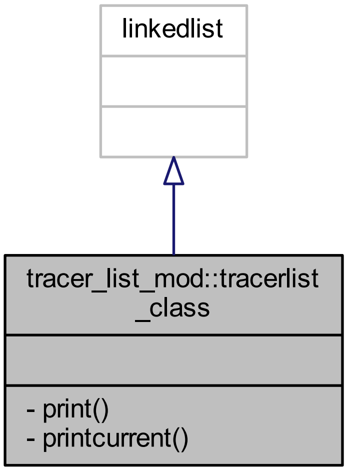
```
50         integer :: ns = mv
```

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

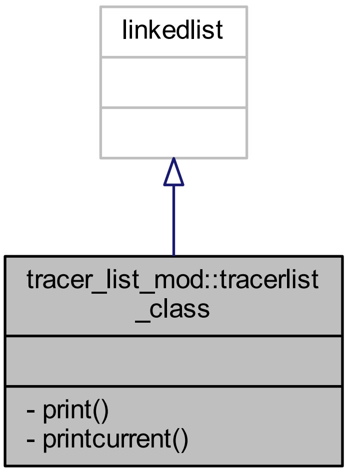
- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracer_base.f90

7.56 tracer_list_mod::tracerlist_class Type Reference

Inheritance diagram for tracer_list_mod::tracerlist_class:



Collaboration diagram for tracer_list_mod::tracerlist_class:



Private Member Functions

- procedure print => print_tracerList
- procedure printcurrent => print_tracerListCurrent

7.56.1 Detailed Description

Definition at line 31 of file tracer_list.f90.

7.56.2 Member Function/Subroutine Documentation

7.56.2.1 print()

```
procedure tracer_list_mod::tracerlist_class::print ( )  [private]
```

Definition at line 33 of file tracer_list.f90.

7.56.2.2 printcurrent()

```
procedure tracer_list_mod::tracerlist_class::printcurrent ( )  [private]
```

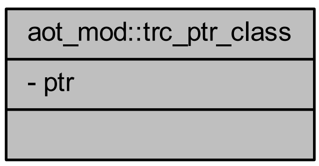
Definition at line 34 of file tracer_list.f90.

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

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

7.57 aot_mod::trc_ptr_class Type Reference

Collaboration diagram for aot_mod::trc_ptr_class:



Private Attributes

- class(tracer_class), pointer ptr
    the actual pointer

7.57.1 Detailed Description

Definition at line 31 of file AoT.f90.

7.57.2 Member Data Documentation

7.57.2.1 ptr

```
class(tracer_class), pointer aot_mod::trc_ptr_class::ptr  [private]
```

the actual pointer

Definition at line 32 of file AoT.f90.

```
32          class(tracer_class), pointer :: ptr
```

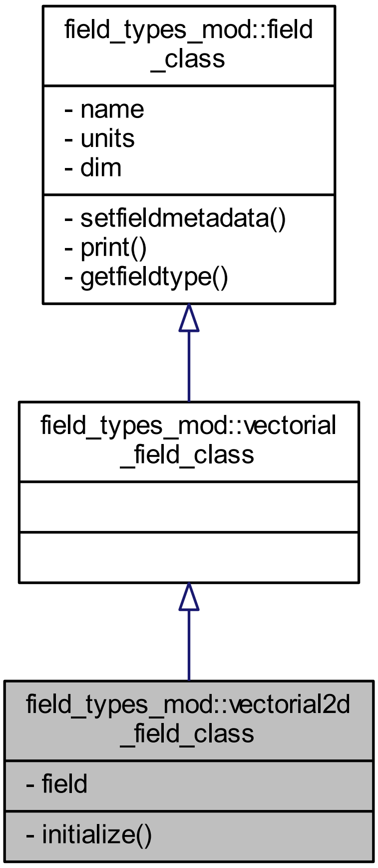
The documentation for this type was generated from the following file:

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

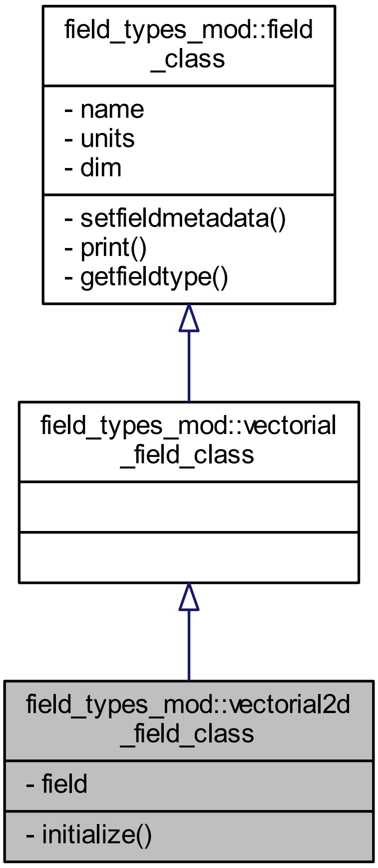
7.58 field_types_mod::vectorial2d_field_class Type Reference

a 2D vectorial field class

Inheritance diagram for field_types_mod::vectorial2d_field_class:



Collaboration diagram for field_types_mod::vectorial2d_field_class:



Private Member Functions

- procedure initialize => initVectorial2dField

Private Attributes

- type(vector), dimension(:,:), allocatable field

    the data on the 2D vectorial data field

7.58.1 Detailed Description

a 2D vectorial field class

Definition at line 84 of file fields_types.f90.

7.58.2 Member Function/Subroutine Documentation

7.58.2.1 initialize()

`procedure field_types_mod::vectorial2d_field_class::initialize ( ) [private]`

Definition at line 87 of file fields_types.f90.

7.58.3 Member Data Documentation

7.58.3.1 field

`type(vector), dimension(:,:), allocatable field_types_mod::vectorial2d_field_class::field [private]`

the data on the 2D vectorial data field

Definition at line 85 of file fields_types.f90.

```
85         type(vector), allocatable, dimension(:,:) :: field
```

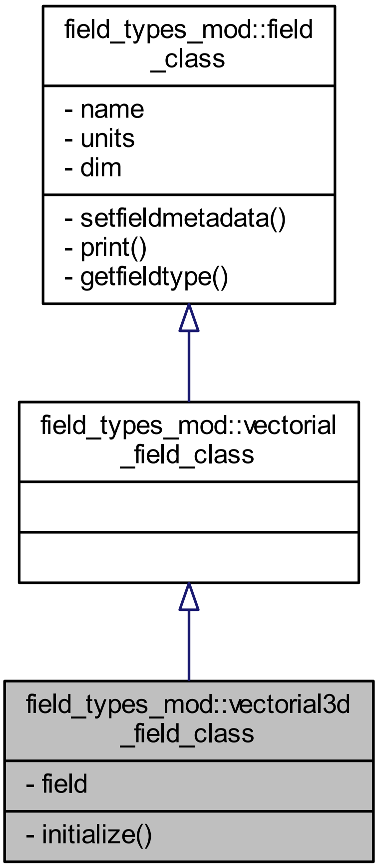
The documentation for this type was generated from the following file:

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

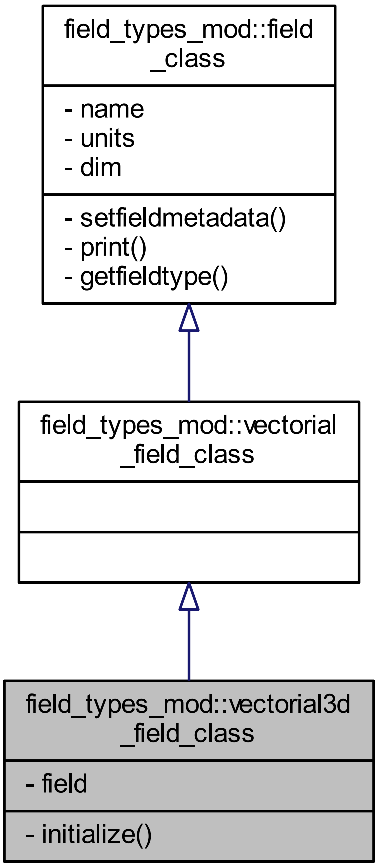
7.59 field_types_mod::vectorial3d_field_class Type Reference

a 3D vectorial field class

Inheritance diagram for field_types_mod::vectorial3d_field_class:



Collaboration diagram for field_types_mod::vectorial3d_field_class:



Private Member Functions

- procedure initialize => initVectorial3dField

Private Attributes

- type(vector), dimension(:,:,:), allocatable field

  the data on the 3D vectorial data field

7.59.1 Detailed Description

a 3D vectorial field class

Definition at line 90 of file fields_types.f90.

7.59.2 Member Function/Subroutine Documentation

7.59.2.1 initialize()

```
procedure field_types_mod::vectorial3d_field_class::initialize ( ) [private]
```

Definition at line 93 of file fields_types.f90.

7.59.3 Member Data Documentation

7.59.3.1 field

```
type(vector), dimension(:,:,:), allocatable field_types_mod::vectorial3d_field_class::field
[private]
```

the data on the 3D vectorial data field

Definition at line 91 of file fields_types.f90.

```
91         type(vector), allocatable, dimension(:,:,:) :: field
```

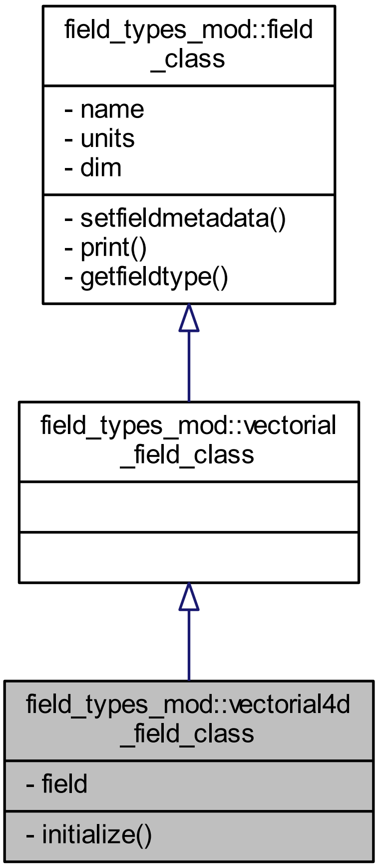
The documentation for this type was generated from the following file:

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

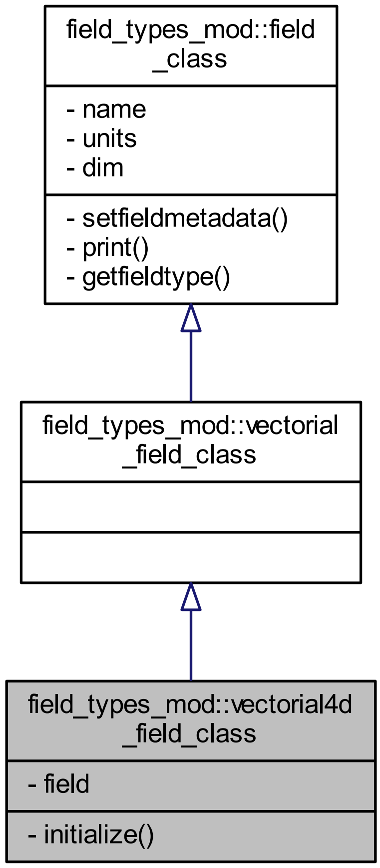
7.60 field_types_mod::vectorial4d_field_class Type Reference

a 4D vectorial field class

Inheritance diagram for field_types_mod::vectorial4d_field_class:



Collaboration diagram for field_types_mod::vectorial4d_field_class:



Private Member Functions

- procedure initialize => initVectorial4dField

Private Attributes

- type(vector), dimension(:,:,:,:), allocatable field

  the data on the 4D vectorial data field

7.60.1 Detailed Description

a 4D vectorial field class

Definition at line 96 of file fields_types.f90.

7.60.2 Member Function/Subroutine Documentation

7.60.2.1 initialize()

```
procedure field_types_mod::vectorial4d_field_class::initialize ( )  [private]
```

Definition at line 99 of file fields_types.f90.

7.60.3 Member Data Documentation

7.60.3.1 field

```
type(vector), dimension(:,:,:,:), allocatable field_types_mod::vectorial4d_field_class::field
[private]
```

the data on the 4D vectorial data field

Definition at line 97 of file fields_types.f90.

```
97        type(vector), allocatable, dimension(:,:,:,:) :: field
```

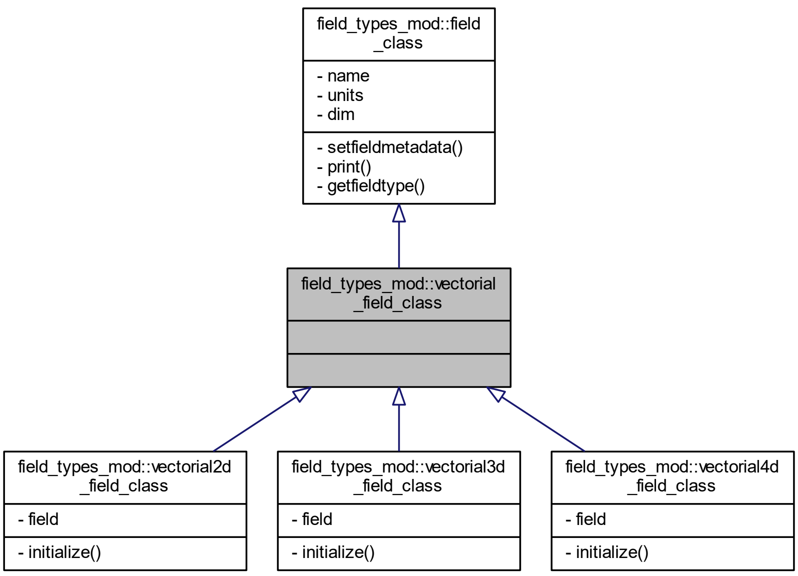
The documentation for this type was generated from the following file:

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

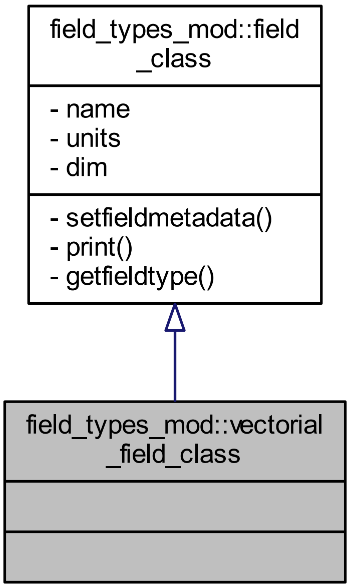
7.61 field_types_mod::vectorial_field_class Type Reference

a vectorial field class

Inheritance diagram for field_types_mod::vectorial_field_class:



Collaboration diagram for field_types_mod::vectorial_field_class:



7.61.1 Detailed Description

a vectorial field class

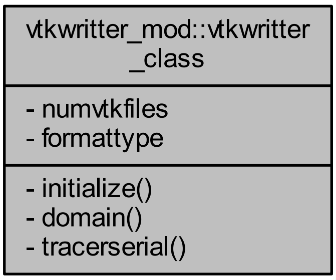
Definition at line 80 of file fields_types.f90.

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

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

7.62 vtkwritter_mod::vtkwritter_class Type Reference

Collaboration diagram for vtkwritter_mod::vtkwritter_class:



Private Member Functions

- procedure initialize => initVTKwritter
- procedure domain
- procedure tracerserial

Private Attributes

- integer numvtkfiles

  number of vtk files written
- type(string) formattype

  format of the data to write on the VTK xml file - ascii, raw, binary

7.62.1 Detailed Description

Definition at line 32 of file vtkwritter.f90.

7.62.2 Member Function/Subroutine Documentation

7.62.2.1 domain()

```
procedure vtkwritter_mod::vtkwritter_class::domain ( )  [private]
```

Definition at line 37 of file vtkwritter.f90.

7.62.2.2 initialize()

```
procedure vtkwritter_mod::vtkwritter_class::initialize ( )  [private]
```

Definition at line 36 of file vtkwritter.f90.

7.62.2.3 tracerserial()

```
procedure vtkwritter_mod::vtkwritter_class::tracerserial ( )  [private]
```

Definition at line 38 of file vtkwritter.f90.

7.62.3 Member Data Documentation

7.62.3.1 formattype

```
type(string) vtkwritter_mod::vtkwritter_class::formattype  [private]
```

format of the data to write on the VTK xml file - ascii, raw, binary

Definition at line 34 of file vtkwritter.f90.

```
34         type(string) :: formatType
```

7.62.3.2 numvtkfiles

```
integer vtkwritter_mod::vtkwritter_class::numvtkfiles  [private]
```

number of vtk files written

Definition at line 33 of file vtkwritter.f90.

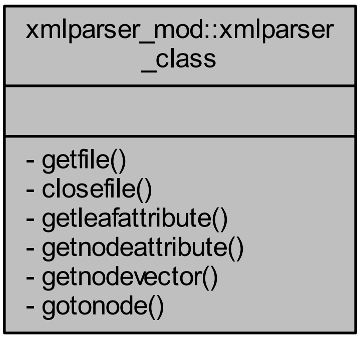
```
33         integer :: numVtkFiles
```

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

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

7.63 xmlparser_mod::xmlparser_class Type Reference

Collaboration diagram for xmlparser_mod::xmlparser_class:



Private Member Functions

- procedure getfile
- procedure closefile
- procedure getleafattribute
- procedure getnodeattribute
- procedure getnodevector
- procedure gotonode

7.63.1 Detailed Description

Definition at line 29 of file xmlparser.f90.

7.63.2 Member Function/Subroutine Documentation

7.63.2.1 closefile()

```
procedure xmlparser_mod::xmlparser_class::closefile ( )  [private]
```

Definition at line 32 of file xmlparser.f90.

7.63.2.2 getfile()

```
procedure xmlparser_mod::xmlparser_class::getfile ( )  [private]
```

Definition at line 31 of file xmlparser.f90.

7.63.2.3 getleafattribute()

```
procedure xmlparser_mod::xmlparser_class::getleafattribute ( )  [private]
```

Definition at line 33 of file xmlparser.f90.

7.63.2.4 getnodeattribute()

```
procedure xmlparser_mod::xmlparser_class::getnodeattribute ( )  [private]
```

Definition at line 34 of file xmlparser.f90.

7.63.2.5 getnodevector()

```
procedure xmlparser_mod::xmlparser_class::getnodevector ( )  [private]
```

Definition at line 35 of file xmlparser.f90.

7.63.2.6 gotonode()

```
procedure xmlparser_mod::xmlparser_class::gotonode ( )  [private]
```

Definition at line 36 of file xmlparser.f90.

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

- C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/xmlparser.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/app/write_vtu.f90 File Reference

Functions/Subroutines

- program write_vtu
- subroutine write_data
- subroutine write_data2
- subroutine write_data3
- subroutine write_data4
- subroutine write_data1

8.3.1 Function/Subroutine Documentation

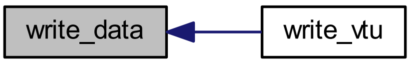
8.3.1.1 write_data()

```
subroutine write_vtu::write_data ( )
```

Definition at line 128 of file write_vtu.f90.

```
128 !------------------------------------------------------------------------------------------------------------
129
130 !------------------------------------------------------------------------------------------------------------
131
132 !------------------------------------------------------------------------------------------------------------
133 error = a_vtk_file%xml_writer%write_piece(np=np, nc=nc)
134 error = a_vtk_file%xml_writer%write_geo(np=np, nc=nc, x=x, y=y, z=z)
135 error = a_vtk_file%xml_writer%write_connectivity(nc=nc, connectivity=connect, offset=offset, cell_type=
    cell_type)
136 error = a_vtk_file%xml_writer%write_dataarray(location='node', action='open')
137 error = a_vtk_file%xml_writer%write_dataarray(data_name='scalars', x=v)
138 error = a_vtk_file%xml_writer%write_dataarray(data_name='vector', x=v_x, y=v_y, z=v_z)
139 error = a_vtk_file%xml_writer%write_dataarray(location='node', action='close')
140 error = a_vtk_file%xml_writer%write_piece()
141 !------------------------------------------------------------------------------------------------------------
```

Here is the caller graph for this function:



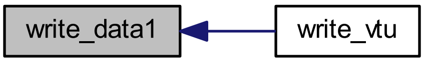
8.3.1.2 write_data1()

```
subroutine write_vtu::write_data1 ( )
```

Definition at line 196 of file write_vtu.f90.

```
196 !------------------------------------------------------------------------------------------------------------
197
198 !------------------------------------------------------------------------------------------------------------
199
200 !------------------------------------------------------------------------------------------------------------
201 error = a_vtk_file%xml_writer%write_piece(np=np1, nc=nc1)
202 error = a_vtk_file%xml_writer%write_geo(np=np1, nc=nc1, x=x1, y=y1, z=z1)
203 error = a_vtk_file%xml_writer%write_connectivity(nc=nc1, connectivity=connect1, offset=offset1, cell_type
    =cell_type1)
204 error = a_vtk_file%xml_writer%write_piece()
205 !------------------------------------------------------------------------------------------------------------
```

Here is the caller graph for this function:



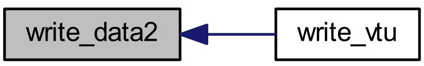
8.3.1.3 write_data2()

```
subroutine write_vtu::write_data2 ( )
```

Definition at line 145 of file write_vtu.f90.

```
145
    !-----------------------------------------------------------------------------------------------------------------------
146
147
    !-----------------------------------------------------------------------------------------------------------------------
148
149
    !-----------------------------------------------------------------------------------------------------------------------
150   error = a_vtk_file%xml_writer%write_piece(np=np2, nc=nc2)
151   error = a_vtk_file%xml_writer%write_geo(np=np2, nc=nc2, x=x2, y=y2, z=z2)
152   error = a_vtk_file%xml_writer%write_connectivity(nc=nc2, connectivity=connect2, offset=offset2, cell_type
      =cell_type2)
153   error = a_vtk_file%xml_writer%write_dataarray(location='node', action='open')
154   error = a_vtk_file%xml_writer%write_dataarray(data_name='scalars', x=v)
155   error = a_vtk_file%xml_writer%write_dataarray(data_name='vector', x=v_x, y=v_y, z=v_z)
156   error = a_vtk_file%xml_writer%write_dataarray(location='node', action='close')
157   error = a_vtk_file%xml_writer%write_piece()
158
    !-----------------------------------------------------------------------------------------------------------------------
```

Here is the caller graph for this function:



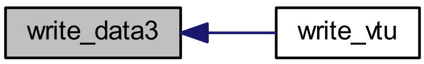
8.3.1.4 write_data3()

```
subroutine write_vtu::write_data3 ( )
```

Definition at line 162 of file write_vtu.f90.

```
162 !------------------------------------------------------------------------------------------------------------
163
164 !------------------------------------------------------------------------------------------------------------
165
166 !------------------------------------------------------------------------------------------------------------
167 error = a_vtk_file%xml_writer%write_piece(np=np3, nc=nc3)
168 error = a_vtk_file%xml_writer%write_geo(np=np3, nc=nc3, x=x3, y=y3, z=z3)
169 error = a_vtk_file%xml_writer%write_connectivity(nc=nc3, connectivity=connect3, offset=offset3, cell_type
    =cell_type3)
170 error = a_vtk_file%xml_writer%write_dataarray(location='node', action='open')
171 error = a_vtk_file%xml_writer%write_dataarray(data_name='scalars', x=v)
172 error = a_vtk_file%xml_writer%write_dataarray(data_name='vector', x=v_x, y=v_y, z=v_z)
173 error = a_vtk_file%xml_writer%write_dataarray(location='node', action='close')
174 error = a_vtk_file%xml_writer%write_piece()
175 !------------------------------------------------------------------------------------------------------------
```

Here is the caller graph for this function:



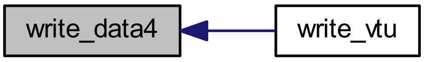
8.3.1.5 write_data4()

```
subroutine write_vtu::write_data4 ( )
```

Definition at line 179 of file write_vtu.f90.

```
179 !------------------------------------------------------------------------------------------------------------
180
181 !------------------------------------------------------------------------------------------------------------
182
183 !------------------------------------------------------------------------------------------------------------
184 error = a_vtk_file%xml_writer%write_piece(np=np4, nc=nc4)
185 error = a_vtk_file%xml_writer%write_geo(np=np4, nc=nc4, x=x4, y=y4, z=z4)
186 error = a_vtk_file%xml_writer%write_connectivity(nc=nc4, connectivity=connect4, offset=offset4, cell_type
    =cell_type4)
187 error = a_vtk_file%xml_writer%write_dataarray(location='node', action='open')
188 error = a_vtk_file%xml_writer%write_dataarray(data_name='scalars', x=v)
189 error = a_vtk_file%xml_writer%write_dataarray(data_name='vector', x=v_x, y=v_y, z=v_z)
190 error = a_vtk_file%xml_writer%write_dataarray(location='node', action='close')
191 error = a_vtk_file%xml_writer%write_piece()
192 !------------------------------------------------------------------------------------------------------------
```

Here is the caller graph for this function:

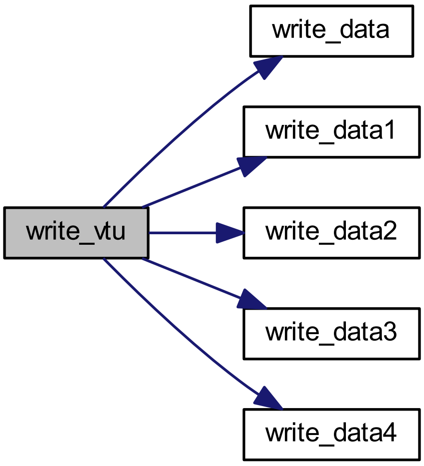


8.3.1.6 write_vtu()

```
program write_vtu ( )
```

Definition at line 2 of file write_vtu.f90.

Here is the call graph for this function:



8.4 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/abstract_LinkedList.f90 File Reference

Data Types

- type abstract_linkedlist_mod::linkedlist

Modules

- module abstract_linkedlist_mod

  Module that defines an unlimited polymorphic container list 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

- subroutine abstract_linkedlist_mod::addvalue (this, value, key)

  Method that stores a value on a new link.
- subroutine abstract_linkedlist_mod::removecurrent (this)

  Method that removes a link from the list.
- subroutine abstract_linkedlist_mod::remove (this, n)

  Method that removes the nth link from a list.
- class(link) function, pointer abstract_linkedlist_mod::getfirst (this)

  Method that returns the first link of the list.
- class(link) function, pointer abstract_linkedlist_mod::getlast (this)

  Method that returns the last link of the list.
- pure integer function abstract_linkedlist_mod::getsize (this)

  Method that returns the size (number of links) of a list.
- class(*) function, pointer abstract_linkedlist_mod::getvalue (this, n)

  Method that returns the value of the nth link of a list.
- class(*) function, pointer abstract_linkedlist_mod::currentvalue (this)

  Method that returns the value of the current link.
- subroutine abstract_linkedlist_mod::next (this)

  Method that returns the next link in the list.
- subroutine abstract_linkedlist_mod::previous (this)

  Method that returns the previous link in the list.
- pure logical function abstract_linkedlist_mod::morevalues (this)

  Method that returns a logical with signaling if the current link is ok.
- subroutine abstract_linkedlist_mod::reset (this)

  Method that resets the list iterator.

8.5 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/AoT.f90 File Reference

Data Types

- type aot_mod::trc_ptr_class
- type aot_mod::aot_class

  Arrays of Tracers class.
- interface aot_mod::aot

Modules

- module aot_mod

  Module to hold the Arrays of Tracers class and its methods. This class defines a collection of id, xyz, uvw, .. arrays that allow for easy and efficient manipulation of the Tracer objects. These must be exported into the objects from this class.

Functions/Subroutines

- type(aot_class) function aot_mod::constructor (trclist)

  Constructor for AoT object with data from a tracerList_class object.
- subroutine aot_mod::clean (self)

  Destructor for AoT object, deallocates all contents.
- subroutine aot_mod::totracers (self)

  Sends the data on the AoT to the Tracer objects. Less type guard checks because they were already made in the constructor of the AoT.
- subroutine aot_mod::print_aot (self)

  Method that prints all the elements of the array.

8.6 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/background.f90 File Reference

Data Types

- type background_mod::fieldslist_class
- type background_mod::background_class
- interface background_mod::background

Modules

- module background_mod

  Defines a background class that describes a solution from which to interpolate. A background object contains an arbitrary number of scalar or vectorial fields, in 2, 3 or 4D, indexed to labeled 1D fields of dimensions. The fields are stored in a linked list, enabling trivial iteration.

Functions/Subroutines

- subroutine background_mod::addfield (self, gfield)

  Method that adds a field to the Background object's field list.
- type(background_class) function background_mod::constructor (id, name, extents, dims)

  Constructor for Background object.
- subroutine background_mod::setdims (self, dims)

  Method that allocates and sets the dimensions of the Background object.
- subroutine background_mod::setextents (self, bbox)

  Method that sets the extents (bounding box) of the Background object.
- subroutine background_mod::setid (self, id, name)

  Method that sets the ID and name of the Background object.
- subroutine background_mod::test (self)

  A class 'unit' test for the background_class.
- subroutine background_mod::printbackground (self)

  Method that prints the Background object.
- subroutine background_mod::print_fieldlist (this)

  Method that prints all the links of the list.
- subroutine background_mod::print_fieldlistcurrent (this)

  Method that prints the current link of the list.

8.7 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

- integer function blocks_mod::numalloctracers (self)

  method that returns the total allocated Tracers in the Block
- subroutine blocks_mod::initblock (self, id, templatebox)

  method to allocate and initialize Blocks and their Emitters
- subroutine blocks_mod::putsource (self, sourcetoadd)

  Method to place a Source on the Block sourceList_class object. Adds the Source info to the Block Emitter.
- subroutine blocks_mod::toogleblocksources (self)

  Method to activate and deactivate the sources on this block, based on GlobaSimTime.
- subroutine blocks_mod::callemitter (self)

  Method to emitt Tracers from currently active Sources on the Block.
- subroutine blocks_mod::distributetracers (self)

  Method to distribute the Tracers to their correct Blocks.
- subroutine blocks_mod::consolidatearrays (self)

  Method to clean the Tracer list from inactive Tracers. TODO test further optimization.
- subroutine blocks_mod::tracerstoaot (self)

  Method to build the AoT object at this timestep for actual numerical work.
- subroutine blocks_mod::aottotracers (self)

  Method to write the data in the AoT back to the Tracer objects in the list.
- subroutine blocks_mod::cleanaot (self)

  Method to clean out the AoT object.
- subroutine blocks_mod::sendtracer (blk, trc)

  Method to send a Tracer from the current Block to another Block.
- integer function, public blocks_mod::getblockindex (pt)

  Returns the index of a Block for a given set of coordinates.
- 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

8.8 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

8.9 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/common_↩ modules.f90 File Reference

Modules

- module common_modules

  Module to hold all of the commonly used base modules.

8.10 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::deletecontent (this)

  Method that deletes the value 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.

8.11 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::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 that space in the Blocks Tracer array
- subroutine emitter_mod::removesource (self, src)

  method to remove from the total emittable particles count a Source
- subroutine class(source_class), intent(inout) emitter_mod::emitt (self, src, trclist)

  method that emitts the Tracers, based on the Sources on this Block Emitter
- subroutine emitter_mod::tracermaker (self, trc, src, p)

  method that calls the corresponding Tracer constructor, depending on the requested type from the emitting Source

8.12 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/fields_types.f90 File Reference

Data Types

- type field_types_mod::field_class
- type field_types_mod::scalar_field_class
  a scalar field class
- type field_types_mod::scalar1d_field_class
  a 1D scalar field class
- type field_types_mod::scalar2d_field_class
  a 2D scalar field class
- type field_types_mod::scalar3d_field_class
  a 3D scalar field class
- type field_types_mod::scalar4d_field_class
  a 4D scalar field class
- type field_types_mod::vectorial_field_class
  a vectorial field class
- type field_types_mod::vectorial2d_field_class
  a 2D vectorial field class
- type field_types_mod::vectorial3d_field_class
  a 3D vectorial field class
- type field_types_mod::vectorial4d_field_class
  a 4D vectorial field class
- type field_types_mod::generic_field_class
  generic field class. This works as a wrapper for a generic initialization routine.

Modules

- module field_types_mod
  Defines classes for 'fields': 1, 2, 3 and 4D labeled data. Valid for both scalar and vectorial (real) data. Defines a generic wrapper for these classes, that abstracts the user from having to choose their data dimensionality or type to create a field.

Functions/Subroutines

- subroutine field_types_mod::inits1d (self, name, units, field)
  Method that allocates and initializes a scalar 1D field in a generic field.
- subroutine field_types_mod::inits2d (self, name, units, field)
  Method that allocates and initializes a scalar 2D field in a generic field.
- subroutine field_types_mod::inits3d (self, name, units, field)
  Method that allocates and initializes a scalar 3D field in a generic field.
- subroutine field_types_mod::inits4d (self, name, units, field)
  Method that allocates and initializes a scalar 4D field in a generic field.
- subroutine field_types_mod::initv2d (self, name, units, field)
  Method that allocates and initializes a vectorial 2D field in a generic field.
- subroutine field_types_mod::initv3d (self, name, units, field)
  Method that allocates and initializes a vectorial 3D field in a generic field.
- subroutine field_types_mod::initv4d (self, name, units, field)

Method that allocates and initializes a vectorial 4D field in a generic field.
- subroutine field_types_mod::initscalar1dfield (self, name, units, dim, field)

  Method that initializes a scalar 1D field.
- subroutine field_types_mod::initscalar2dfield (self, name, units, dim, field)

  Method that initializes a scalar 2D field.
- subroutine field_types_mod::initscalar3dfield (self, name, units, dim, field)

  Method that initializes a scalar 3D field.
- subroutine field_types_mod::initscalar4dfield (self, name, units, dim, field)

  Method that initializes a scalar 4D field.
- subroutine field_types_mod::initvectorial2dfield (self, name, units, dim, field)

  Method that initializes a vectorial 2D field.
- subroutine field_types_mod::initvectorial3dfield (self, name, units, dim, field)

  Method that initializes a vectorial 3D field.
- subroutine field_types_mod::initvectorial4dfield (self, name, units, dim, field)

  Method that initializes a vectorial 4D field.
- subroutine field_types_mod::setfieldmetadata (self, name, units, dim)

  Method that initializes a base field object by filling metadata.
- subroutine field_types_mod::printgenericfield (self)

  Method that prints the generic field information.
- subroutine field_types_mod::test (self)

  A class 'unit' test for the generic_field_class.
- subroutine field_types_mod::printfield (self)

  Method that prints the field information.
- type(string) function field_types_mod::getfieldtype (self)

  Method that returns the field type (scalar or vectorial), in a string.

8.13 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, dp)

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

  method to get the list of points that fill a given geometry
- type(vector) function geometry_mod::getcenter (self, shapetype)

  method to get the baricenter of a given geometry
- type(vector) function, dimension(:), allocatable geometry_mod::getpoints (self, shapetype)

  method that returns the points defining a given geometry
- integer function geometry_mod::getnumpoints (self, shapetype)

  method the points defining 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

8.14 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/link.f90 File Reference

Data Types

- interface link_mod::link
- interface link_mod::link

Modules

- module link_mod

  Module that defines a link based on an unlimited polymorphic container class.

Functions/Subroutines

- class(∗) function, pointer link_mod::getvalue (this)

  Method that returns a pointer to the values stored in the container in this link.
- class(link) function, pointer link_mod::nextlink (this)

  Method that returns a pointer to the next link in a list.
- class(link) function, pointer link_mod::previouslink (this)

  Method that returns a pointer to the previous link in a list.
- subroutine link_mod::setnextlink (this, next)

  Method to set the next link in a list.
- subroutine link_mod::setpreviouslink (this, prev)

  Method to set the previous link in a list.
- subroutine link_mod::removelink (this)

  Method to remove a link in a list.
- class(link) function, pointer link_mod::constructor (to_store, prev, next, key)

  Link constructor, can be used with the 'link' name since it was defined as such in an interface declaration.

8.15 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. This is the only class that is exposed to an external program, as it encapsulates every other class and method.

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::togglesources (self)

  Simulation method to activate and deactivate Sources based on the GlobalSimTime.
- subroutine simulation_mod::blocksemitt (self)

  Simulation method to call the Blocks to emitt tracers at current SimTime.
- subroutine simulation_mod::blocksdistribute (self)

  Simulation method to call the Blocks to distribute Tracers at current SimTime.
- subroutine simulation_mod::blocksconsolidatearrays (self)

  Simulation method to call the Blocks to consolidate the Tracer array at current SimTime.
- subroutine simulation_mod::blockstracerstoaot (self)

  Simulation method to call the Blocks to build their Array of Tracers (AoT) from the Tracer list at current SimTime.
- subroutine simulation_mod::blocksaottotracers (self)

Simulation method to call the Blocks to print their Array of Tracers (AoT) back to the Tracer objects on the list at current SimTime.
- subroutine simulation_mod::blockscleanaot (self)

  Simulation method to call the Blocks to clean their Array of Tracers (AoT) at current SimTime.
- subroutine simulation_mod::setinitialstate (self)

  Simulation method to distribute the Sources to the Blocks, allocate the respective Tracers and redistribute if needed.
- integer function simulation_mod::gettracertotals (self)

  Simulation method to count Tracer numbers.
- subroutine simulation_mod::printtracertotals (self)

  Simulation method to count Tracer numbers.
- subroutine simulation_mod::settracermemory (self, ntrc)

  Simulation method to account for Tracer memory consumption.
- 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.

8.16 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/simulation_about.f90 File Reference

Modules

- module simulation_about_mod

  Module to print version, licence, preambles.

Functions/Subroutines

- subroutine, public simulation_about_mod::printlicpreamble

  Public licence and preamble printer routine.

Variables

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

8.17 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::src_parm_t

  Lists for Source parameters.
- type simulation_globals_mod::sim_t

  Simulation related counters and others.
- 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, outpath)

  Globals default setting routine.
- subroutine simulation_globals_mod::increment_numtracer (self)

  Increments Tracer count. This routine MUST be ATOMIC.
- integer function simulation_globals_mod::getnumtracer (self)

  Returns a new ID for a Tracer.
- subroutine simulation_globals_mod::increment_numdt (self)

  incrementing time step count.
- integer function simulation_globals_mod::getnumdt (self)

  Returns the number of time steps.
- subroutine simulation_globals_mod::increment_numoutfile (self)

  incrementing output file count.
- integer function simulation_globals_mod::getnumoutfile (self)

  Returns the number of output files written.
- subroutine simulation_globals_mod::buildlists (self)

  Method to build the parameters list of the Sources.
- 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::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

8.18 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/simulation_initialize_mod.f90 File Reference

Modules

- module simulation_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 simulation_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 simulation_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 simulation_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 simulation_initialize_mod::init_sources (case_node)

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

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

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

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

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

8.19 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

8.20 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)
  Memory logger initialization method.
- subroutine simulation_memory_mod::getotal (self, size)
  Method to retreive the total size of the allocated memory.
- subroutine simulation_memory_mod::setntrc (self, ntrc)
  Method to set the total expected number of Tracers.
- subroutine simulation_memory_mod::setsizetrc (self, sizeTrc)
  Method to set the size of a typical Tracer.
- subroutine simulation_memory_mod::addblock (self, size)
  Method to add the size of a Block to the memory log.
- subroutine simulation_memory_mod::addsource (self, size)
  Method to add the size of a Source to the memory log.
- subroutine simulation_memory_mod::setracer (self, size)
  Method to add the size of a Tracer to the memory log.
- subroutine simulation_memory_mod::adddef (self, size)
  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)
  Method to print the allocated memory.

Variables

- type(memory_t), public simulation_memory_mod::simmemory

8.21 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/simulation_↩ output_streamer.f90 File Reference

Data Types

- type simulation_output_streamer_mod::output_streamer_class

Modules

- module simulation_output_streamer_mod

  Defines a output file writer class with an object exposable to the Simulation This class is in charge of selectig the correct writter for the selected output file format.

Functions/Subroutines

- subroutine simulation_output_streamer_mod::initoutputstreamer (self)

  Initializes the Output writer object.
- subroutine simulation_output_streamer_mod::writestepserial (self, blocks)

  Streamer method to call a simulation step writer. Writes binary XML VTK format using an unstructured grid.
- subroutine simulation_output_streamer_mod::writedomain (self, filename, bbox, npbbox, blocks)

  Public simulation domain writting routine. Writes binary XML VTK format using an unstructured grid.

Variables

- type(output_streamer_class), public simulation_output_streamer_mod::outputstreamer

8.22 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::sps = kind(1._R4P)

  Simple precision definition switch.
- integer, parameter simulation_precision_mod::dps = kind(1._R8P)

  Double precision definition switch.
- integer, parameter, public simulation_precision_mod::prec = dps
- integer, parameter, public simulation_precision_mod::prec_time = sps
- integer, parameter, public simulation_precision_mod::prec_wrt = sps
- real(prec), parameter, public simulation_precision_mod::missing_value_default = -9999.0_dps
- 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_dps
- integer, parameter, public simulation_precision_mod::err_ind = -1
- integer, parameter, public simulation_precision_mod::char_len = 99

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

Data Types

- type sources_mod::source_par
- type sources_mod::source_prop

  Type - material properties of a source object.
- 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::linkproperty (src, ptype, pname)

  source property setting proceadure - initializes Source variables
- subroutine sources_mod::setpropertynames (self, srcid_str, ptype, pname)

  source property setting routine, calls source by id to set its properties
- subroutine sources_mod::setpropertyatributes (src, pname, pvalue)

  source property atribute setting proceadure - initializes Source variables
- subroutine sources_mod::check (self)

  Method that checks for the consistency of the Source properties.
- subroutine sources_mod::initializesource (src, id, name, emitting_rate, start, finish, source_geometry, shapetype)

  source inititialization proceadure - initializes Source variables
- logical function sources_mod::isparticulate (self)

  Returns particulate status of this Source, i.e, true if the emitted Tracers are actually a collection of particles with an evolving concentration.
- subroutine sources_mod::setotalnp (self)

  method that sets the total number of tracers a source will potentially create
- subroutine sources_mod::printsource (src)

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

Variables

- type(source_group_class), public sources_mod::tempsources

  Temporary Source array, used exclusively for building the case from a description file.

8.24 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/sources_list.f90 File Reference

Data Types

- type sources_list_mod::sourcelist_class

Modules

- module sources_list_mod

  Module to hold the Sources linked list class and its methods. This class defines a double linked list to store any variable type, but with specific methods with type guards for Source objects. The class allows for insertion, deletion and iteration of the desired contents.

Functions/Subroutines

- subroutine sources_list_mod::print_sourcelist (this)

  Method that prints all the links of the list.
- subroutine sources_list_mod::print_sourcelistcurrent (this)

  Method that prints the current link of the list.

8.25 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.
- interface tracer_base_mod::tracer

Modules

- module tracer_base_mod

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

Functions/Subroutines

- subroutine tracer_base_mod::printtracer (self)

  Method to print basic info about the Tracer.
- type(tracer_class) function tracer_base_mod::constructor (id, src, time, p)

  Base Tracer constructor.

Variables

- type(tracer_class), public tracer_base_mod::dummytracer

  Just a template to allocate the generic arrays to this size.

8.26 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/tracer_list.f90 File Reference

Data Types

- type tracer_list_mod::tracerlist_class

Modules

- module tracer_list_mod

  Module to hold the tracer linked list class and its methods. This class defines a double linked list to store any variable type, but with specific methods with type guards for Tracer objects. The class allows for insertion, deletion and iteration of the desired contents.

Functions/Subroutines

- subroutine tracer_list_mod::print_tracerlist (this)

  Method that prints all the links of the list.
- subroutine tracer_list_mod::print_tracerlistcurrent (this)

  Method that prints the current link of the list.

8.27 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.
- interface tracer_paper_mod::papertracer

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

- type(paper_class) function tracer_paper_mod::constructor (id, src, time, p)

  Paper Tracer constructor.

8.28 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.

8.29 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.

8.30 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

- type(vector) function, public utilities_mod::geo2m (geovec, lat)

    Public function that returns a vector in meters given an array in geographical coordinates (lon, lat, z) and a lattitude.
- type(vector) function, public utilities_mod::m2geo (mvec, lat)

    Public function that returns a vector in geographical coordinates (lon, lat, z) given an array in meters and a lattitude.
- character(:) function, allocatable, public utilities_mod::int2str (fmt, i)

    Public function that returns a zero paded string from an integer number and a format descriptor.
- real(prec) function, public utilities_mod::get_closest_twopow (num)

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

8.31 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/vtkwritter.f90 File Reference

Data Types

- type vtkwritter_mod::vtkwritter_class

Modules

- module vtkwritter_mod

  Defines a vtk writer class with an object exposable to the Output streamer. Writes files in .xml vtk, both in serial and parallel model. Uses an unstructured mesh format specifier to store any type of data, both meshes and Tracers. Supports scalar and vectorial data.

Functions/Subroutines

- subroutine vtkwritter_mod::initvtkwritter (self)

  Initializes a VTK writer object.
- subroutine vtkwritter_mod::tracerserial (self, filename, blocks)

  Public Tracer writting routine. Writes Tracer data in binary XML VTK format using an unstructured grid. Serial writer for serial files.
- subroutine vtkwritter_mod::domain (self, filename, bbox, npbbox, blocks)

  Public simulation domain writting routine. Writes binary XML VTK format using an unstructured grid.

Variables

- type(vtkwritter_class), public vtkwritter_mod::vtkwritter

8.32 C:/Users/administrator/Documents/GitHub/MOHID-Lagrangian/src/lib/xmlparser.f90 File Reference

Data Types

- type xmlparser_mod::xmlparser_class

Modules

- module xmlparser_mod

  Module with the simulation xml parsing class and methods, Encapsulates the FOX_dom library.

Functions/Subroutines

- subroutine xmlparser_mod::getfile (self, xmldoc, xmlfilename)

  Method that parses an xml file and returns a pointer to the master node.
- subroutine xmlparser_mod::closefile (self, xmldoc)

  Method that closes a parsed xml file or node.
- subroutine xmlparser_mod::getleafattribute (self, xmlnode, att_name, att_value)

  Method that parses an xml attribute. Reads the requested attribute from a given leaf node,.
- subroutine xmlparser_mod::getnodeattribute (self, xmlnode, tag, att_name, att_value, read_flag, mandatory)

  Method that parses an attribute from an xml node. In the format '<Tag att_name="att_value">'.
- subroutine xmlparser_mod::getnodevector (self, xmlnode, tag, vec, read_flag, mandatory)

  Method to parse xyz vectors in xml files. Vector must be in format '<Tag x="vec%x" y="vec%y" z="vec%z">'.
- subroutine xmlparser_mod::gotonode (self, currentNode, targetNode, targetNodeName, read_flag, mandatory)

  Method that retrieves a node from within a node. Returns a nullifyed pointer if not found, stops if mandatory.

Variables

- type(xmlparser_class), public xmlparser_mod::xmlreader