

---

# **Earth Diagnostics Documentation**

***Release 3.0.0b3***

**BSC-CNS Earth Sciences Department**

Jul 18, 2016



## CONTENTS

<b>1</b>	<b>Tutorial</b>	<b>1</b>
1.1	Installation . . . . .	1
1.2	Creating a config file . . . . .	1
<b>2</b>	<b>Tips and tricks</b>	<b>3</b>
2.1	Working with ORCA1 . . . . .	3
2.2	Configuring core usage . . . . .	3
2.3	NEMO files . . . . .	3
<b>3</b>	<b>What to do if you have an error</b>	<b>5</b>
<b>4</b>	<b>Developer's guide</b>	<b>7</b>
4.1	Developing a diagnostic . . . . .	7
<b>5</b>	<b>Frequently Asked Questions</b>	<b>9</b>
<b>6</b>	<b>Module documentation</b>	<b>11</b>
6.1	earthdiagnostics . . . . .	11
6.1.1	earthdiagnostics.box . . . . .	11
6.1.2	earthdiagnostics.constants . . . . .	12
6.1.3	earthdiagnostics.cdftools . . . . .	16
6.1.4	earthdiagnostics.datamanager . . . . .	16
6.1.5	earthdiagnostics.diagnostic . . . . .	19
6.1.6	earthdiagnostics.diagnostics . . . . .	20
6.1.7	earthdiagnostics.experimentmanager . . . . .	20
6.1.8	earthdiagnostics.utils . . . . .	21
6.2	earthdiagnostics.ocean . . . . .	23
6.2.1	earthdiagnostics.ocean.areamoc . . . . .	23
6.2.2	earthdiagnostics.ocean.averagesection . . . . .	24
6.2.3	earthdiagnostics.ocean.convectionsites . . . . .	25
6.2.4	earthdiagnostics.ocean.cutsection . . . . .	26
6.2.5	earthdiagnostics.ocean.gyres . . . . .	26
6.2.6	earthdiagnostics.ocean.interpolate . . . . .	27
6.2.7	earthdiagnostics.ocean.maxmoc . . . . .	28
6.2.8	earthdiagnostics.ocean.mixedlayersaltcontent . . . . .	29
6.2.9	earthdiagnostics.ocean.moc . . . . .	29
6.2.10	earthdiagnostics.ocean.psi . . . . .	30
6.2.11	earthdiagnostics.ocean.siasiesiv . . . . .	31
6.2.12	earthdiagnostics.ocean.verticalmean . . . . .	31
6.2.13	earthdiagnostics.ocean.verticalmeanmeters . . . . .	32

<b>Python Module Index</b>	<b>35</b>
<b>Index</b>	<b>37</b>

## TUTORIAL

So, you are planning to use the Earth Diagnostics? You don't know how to use them? This is the place to go. From now on this tutorial will guide you through all the process from installation to running.

---

**Hint:** If you have any problem with this tutorial, please report it to <[javier.vegas@bsc.es](mailto:javier.vegas@bsc.es)> so it can be corrected. A lot of people will benefit from it and you will be mentioned here.

---

### 1.1 Installation

For now, you only have an option: download the diagnostics directly from BSC-ES's Gitlab:

```
git clone https://earth.bsc.es/gitlab/es/ocean_diagnostics.git
```

You will also need

- CDO version 1.6.9 (other versions could work, but this is the one we use)
- NCO version 4.5.4 or newer
- Python 2.7 or newer (but no 3.x) with Autosubmit, CDO and NCO packages
- Access to CDFTOOLS\_3.0 executables for BSC-ES. At this point, those are located at /home/Earth/jvegas/CDFTOOLS\_CMOR/bin.

Call the diags with -h option to see if everything is ready:

```
${PATH_TO_REPOSITORY}/earthdiagnostics.diags.py -h
```

### 1.2 Creating a config file

If you go into the earthdiagnostics folder in the git repository, you will see a diags.conf that can be used as a model for your config file. It contains commentaries explaining what represents each one of its parameters, so please read it carefully.

Once you have configured your experiment you can execute any diagnostic by calling this command (substitute the variables for the real paths before launching, please)

```
${PATH_TO_REPOSITORY}/earthdiagnostics.diags.py ${PATH_TO_MY_CONF_FILE}
```

And... that's it. You will find your results inside CMOR's folder tree and a folder for the temp files in the scratch. This folder is named after the EXPID.



## TIPS AND TRICKS

### 2.1 Working with ORCA1

If you plan to run diagnostics for ORCA1 resolution, be aware that your workstation will be more than capable to run them. At this resolution, memory and time consumption is low enough to allow you keep using the machine while running, specially if you reserve a pair of cores for other uses.

### 2.2 Configuring core usage

By default, the Earth Diagnostics creates a thread for each available core for the execution. If you are using a queueing system, the diagnostics will always use the number of cores that you reserved. If you are running outside a queueing system, the diagnostics will try to use all the cores on the machine. To avoid this, add the `MAX_CORES` parameter to the `DIAGNOSTICS` section inside the `diags.conf` file that you are using.

### 2.3 NEMO files

Unlike the bash version of the ocean diagnostics, this program keeps the NEMO files in the scratch folder so you can launch different configurations for the same experiment with reduced start time. You will need to remove the experiment's folder in the scratch directory at the end of the experiment to avoid wasting resources.





## WHAT TO DO IF YOU HAVE AN ERROR

Sometimes, the diagnostics may crash and you will not know why. This section will give you a procedure to follow before reporting the issue. This procedure is intended to solve some common problems or, at least, to help you in creating good issue reports. Remember: a good issue report reduces the time required to solve it!

---

**Hint:** Reading is good. Most times the error message will point you to the problem's source and sometimes even give you a hint of how to solve it by yourself.

---

Try this simple steps BEFORE reporting an issue

- Clean scratch folder
- Update to the latest compatible tag: maybe your issue is already solved in it
- If you get the error for the first chunk of a given diagnostic, change the number of chunks to 1
- Call the diags with the `-lc DEBUG -log log.txt` options

Now, you have two options: if everything is fine, the error was probably due to some corrupted files or some unstable machine state. Nevertheless, try running the diagnostic with `-lc DEBUG -log log.txt` for all the chunks. If everything it's fine that's all.

If you experienced the same problem again, go to the GitLab portal and look into the open issues ( [https://earth.bsc.es/gitlab/es/ocean\\_diagnostics/issues](https://earth.bsc.es/gitlab/es/ocean_diagnostics/issues) ). If you find your issue or a very similar one, use it to report your problems. If you can not find an open one that suites your problem, create a new one and explain what is happening to you.

In any case, it will be very useful if you can attach your `diags.conf` and `log.txt` files.

After that, it's just a matter of waiting for the developers to do their work and answering the questions that they may have. Please, be patient.



## DEVELOPER'S GUIDE

The tool provides a set of useful diagnostics, but a lot more can be required at anytime. If you miss something and are able to develop it, you are more than welcome to collaborate. Even if you can not develop it, please let us know what do you want.

The first step is to go to the GitLab page for the project ( [https://earth.bsc.es/gitlab/es/ocean\\_diagnostics/](https://earth.bsc.es/gitlab/es/ocean_diagnostics/) ) and open a new issue. Be sure that the title is self-explicative and give a detailed description of what you want. Please, be very explicit about what you want to avoid misunderstandings.

---

**Hint:** If reading your description, you think that you are taking the developers as stupids, you are doing it perfectly.

---

Don't forget to add the relevant tags. At this stage you will have to choose between 'enhancement', if you are proposing an improvement on a currently available feature, or 'new feature' in any the other case.

Now, if you are thinking on developing it yourself, please refer to the BSC-ES Git strategy ( [wiki\\_link\\_when\\_available](#) ) If you have any doubts, or just want help to start the development, contact [javier.vegas@bsc.es](mailto:javier.vegas@bsc.es).

### 4.1 Developing a diagnostic

For new diagnostics development, we have some advice to give:

- Do not worry about performance at first, just create a version that works. Developers can help you to optimize it later.
- There is nothing wrong with doing some common preparations in the `generate_jobs` of the diagnostic.
- Parallelization is achieved by running multiple diagnostics at a time. You don't need to implement it at diagnostic level
- Use the smallest time frame for your diagnostic: if you can work at chunk level, do not ask for full year data.
- Prefer NCO over CDO, you will have less problems when versions change.
- Ask for help as soon as you get stuck.
- Use always the methods in Utils instead of writing your own code.
- Use meaningful variable names. If you are using short names just to write less, please switch to an editor with autocompletion!
- Do not modify the mesh and mask files, another diagnostic can be using them at the same time.



## **FREQUENTLY ASKED QUESTIONS**

Here will be the answers to the most usual questions. For the moment, there is nothing to see here...



---

MODULE DOCUMENTATION

## 6.1 earthdiagnostics

### 6.1.1 earthdiagnostics.box

**class** earthdiagnostics.box.**Box** (*depth\_in\_meters=False*)

Bases: `object`

Represents a box in the 3D space. Also allows easy conversion from the coordinate values to significant string representations

**depth\_in\_meters = None**

If True, treats the depth as if it is given in meters. If False, as it is given in levels :rtype: bool

**get\_depth\_str()**

Gets a string representation of depth. For depth expressed in meters, it adds th character 'm' to the end  
If min\_depth is different from max\_depth, it concatenates the two values :return: string representation for depth :rtype: str

**get\_lat\_str()**

Gets a string representation of the latitude in the format XX{N/S}. If min\_lat is different from max\_lat, it concatenates the two values :return: string representation for latitude :rtype: str

**get\_lon\_str()**

Gets a string representation of the longitude in the format XX{E/W}. If min\_lon is different from max\_lon, it concatenates the two values :return: string representation for longitude :rtype: str

**max\_depth = None**

Maximum depth :rtype: float

**max\_lat**

Maximum latitude :rtype: float

**max\_lon**

Maximum longitude :rtype: float

**min\_depth = None**

Minimum depth :rtype: float

**min\_lat**

Minimum latitude :rtype: float

**min\_lon**

Minimum longitude :rtype: float

## 6.1.2 earthdiagnostics.constants

Contains the enumeration-like classes used by the diagnostics

**class** `earthdiagnostics.constants.Basin` (*shortname, fullname, box=None*)

Bases: `object`

Class representing a given basin

### Parameters

- **shortname** (*str*) – sfull basin's name
- **fullname** (*str*) – full basin's name
- **box** (`Box`) – box defining the basin

**box = None**

Box representing the basin

**fullname**

Basin's full name :rtype: str

**shortname**

Basin's short name :rtype: str

**class** `earthdiagnostics.constants.Basins`

Bases: `object`

Predefined basins

**Antarctic** = `<earthdiagnostics.constants.Basin object>`

Antarctic Ocean

**AntarcticAtlantic** = `<earthdiagnostics.constants.Basin object>`

Antarctic Ocean Atlantic Sector

**AntarcticIndian** = `<earthdiagnostics.constants.Basin object>`

Antarctic Ocean Indian Sector

**Arctic** = `<earthdiagnostics.constants.Basin object>`

Arctic Ocean

**ArcticMarginalSeas** = `<earthdiagnostics.constants.Basin object>`

Arctic Ocean

**ArcticNorthAtlantic** = `<earthdiagnostics.constants.Basin object>`

Arctic Ocean North Atlantic

**Atlantic** = `<earthdiagnostics.constants.Basin object>`

Atlantic ocean

**Baffin** = `<earthdiagnostics.constants.Basin object>`

Baffin

**Baffin\_Bay** = `<earthdiagnostics.constants.Basin object>`

Baffin\_Bay

**Baltic\_Sea** = `<earthdiagnostics.constants.Basin object>`

Baltic\_Sea

**BarKara** = `<earthdiagnostics.constants.Basin object>`

BarKara



**Barents\_Sea** = <earthdiagnostics.constants.Basin object>  
Barents\_Sea

**Beaufort\_Chukchi\_Sea** = <earthdiagnostics.constants.Basin object>  
Beaufort\_Chukchi\_Sea

**Beaufort\_Sea** = <earthdiagnostics.constants.Basin object>  
Beaufort\_Sea

**Bellingshausen\_Sea** = <earthdiagnostics.constants.Basin object>  
Bellingshausen\_Sea

**Bering** = <earthdiagnostics.constants.Basin object>  
Bering

**Bering\_Strait** = <earthdiagnostics.constants.Basin object>  
Bering\_Strait

**CanArch** = <earthdiagnostics.constants.Basin object>  
CanArch

**Canadian\_Waters** = <earthdiagnostics.constants.Basin object>  
Canadian\_Waters

**Caspian\_Sea** = <earthdiagnostics.constants.Basin object>  
Caspian\_Sea

**Central\_Arctic** = <earthdiagnostics.constants.Basin object>  
Central\_Arctic

**Chukchi\_Sea** = <earthdiagnostics.constants.Basin object>  
Chukchi\_Sea

**East\_Siberian\_Sea** = <earthdiagnostics.constants.Basin object>  
East\_Siberian\_Sea

**Eastern\_Central\_Arctic** = <earthdiagnostics.constants.Basin object>  
Eastern\_Central\_Arctic

**Fram\_Strait** = <earthdiagnostics.constants.Basin object>  
Fram\_Strait

**Global** = <earthdiagnostics.constants.Basin object>  
Global ocean

**Global\_Ocean** = <earthdiagnostics.constants.Basin object>  
Global\_Ocean

**Greenland\_Sea** = <earthdiagnostics.constants.Basin object>  
Greenland\_Sea

**Grnland** = <earthdiagnostics.constants.Basin object>  
Grnland

**Hudson** = <earthdiagnostics.constants.Basin object>  
Hudson

**Icelandic\_Sea** = <earthdiagnostics.constants.Basin object>  
Icelandic\_Sea

**Indian** = <earthdiagnostics.constants.Basin object>  
Indian Ocean

**IndoPacific** = <earthdiagnostics.constants.Basin object>  
Indo Pacific Ocean

**Kara\_Gate\_Strait** = <earthdiagnostics.constants.Basin object>  
Kara\_Gate\_Strait

**Kara\_Sea** = <earthdiagnostics.constants.Basin object>  
Kara\_Sea

**Labrador\_Sea** = <earthdiagnostics.constants.Basin object>  
Labrador\_Sea

**Laptev\_East\_Siberian\_Chukchi\_Seas** = <earthdiagnostics.constants.Basin object>  
Laptev\_East\_Siberian\_Chukchi\_Seas

**Laptev\_East\_Siberian\_Seas** = <earthdiagnostics.constants.Basin object>  
Laptev\_East\_Siberian\_Seas

**Laptev\_Sea** = <earthdiagnostics.constants.Basin object>  
Laptev\_Sea

**Lincoln\_Sea** = <earthdiagnostics.constants.Basin object>  
Lincoln\_Sea

**Mediterranean\_Sea** = <earthdiagnostics.constants.Basin object>  
Mediterranean\_Sea

**Nares\_Strait** = <earthdiagnostics.constants.Basin object>  
Nares\_Strait

**Nordic\_Barents\_Seas** = <earthdiagnostics.constants.Basin object>  
Nordic\_Barents\_Seas

**Nordic\_Seas** = <earthdiagnostics.constants.Basin object>  
Nordic\_Seas

**NorthAtlantic** = <earthdiagnostics.constants.Basin object>  
North Atlantic Ocean

**NorthPacific** = <earthdiagnostics.constants.Basin object>  
North Pacific Ocean

**NorthWest\_Passage** = <earthdiagnostics.constants.Basin object>  
NorthWest\_Passage

**North\_Atlantic\_Arctic** = <earthdiagnostics.constants.Basin object>  
North\_Atlantic\_Arctic

**North\_Hemisphere\_Ocean** = <earthdiagnostics.constants.Basin object>  
North\_Hemisphere\_Ocean

**Norwegian\_Sea** = <earthdiagnostics.constants.Basin object>  
Norwegian\_Sea

**Okhotsk** = <earthdiagnostics.constants.Basin object>  
Okhotsk

**OpenOcean** = <earthdiagnostics.constants.Basin object>  
OpenOcean

**Pacific** = <earthdiagnostics.constants.Basin object>  
Pacific Ocean

**Ross\_Sea** = <earthdiagnostics.constants.Basin object>  
 Ross\_Sea

**Serreze\_Arctic** = <earthdiagnostics.constants.Basin object>  
 Serreze\_Arctic

**Southern\_Hemisphere** = <earthdiagnostics.constants.Basin object>  
 Southern\_Hemisphere

**StLawr** = <earthdiagnostics.constants.Basin object>  
 StLawr

**Subpolar\_Gyre** = <earthdiagnostics.constants.Basin object>  
 Subpolar\_Gyre

**TotalArc** = <earthdiagnostics.constants.Basin object>  
 TotalArc

**TropicalAtlantic** = <earthdiagnostics.constants.Basin object>  
 Tropical Atlantic Ocean

**TropicalIndian** = <earthdiagnostics.constants.Basin object>  
 Tropical Indian Ocean

**TropicalPacific** = <earthdiagnostics.constants.Basin object>  
 Tropical Pacific Ocean

**Vilkitsky\_Strait** = <earthdiagnostics.constants.Basin object>  
 Vilkitsky\_Strait

**Weddell\_Sea** = <earthdiagnostics.constants.Basin object>  
 Weddell\_Sea

**Western\_Central\_Arctic** = <earthdiagnostics.constants.Basin object>  
 Western\_Central\_Arctic

**classmethod parse** (*basin*)

Return the basin matching the given name. If the parameter *basin* is a *Basin* instance, directly returns the same instance. This behaviour is intended to facilitate the development of methods that can either accept a name or a *Basin* instance to characterize the basin.

**Parameters** **basin** (*str* | *Basin*) – basin name or basin instance

**Returns** basin instance corresponding to the basin name

**Return type** *Basin*

**class** earthdiagnostics.constants.**Models**

Bases: *object*

Predefined models

**ECEARTH\_2\_3\_O1L42** = 'Ec2.3\_O1L42'  
 EC-Earth 2.3 ORCA1 L42

**ECEARTH\_3\_0\_O1L46** = 'Ec3.0\_O1L46'  
 EC-Earth 3 ORCA1 L46

**ECEARTH\_3\_0\_O25L46** = 'Ec3.0\_O25L46'  
 EC-Earth 3 ORCA0.25 L46

**ECEARTH\_3\_0\_O25L75** = 'Ec3.0\_O25L75'  
 EC-Earth 3 ORCA0.25 L75

**ECEARTH\_3\_1\_O25L75** = 'Ec3.1\_O25L75'

EC-Earth 3.1 ORCA0.25 L75

**GLORYS2\_V1\_O25L75** = 'glorys2v1\_O25L75'

GLORYS2v1 ORCA0.25 L75

**NEMOVAR\_O1L42** = 'nemovar\_O1L42'

NEMOVAR ORCA1 L42

**NEMO\_3\_2\_O1L42** = 'N3.2\_O1L42'

NEMO 3.2 ORCA1 L42

**NEMO\_3\_3\_O1L46** = 'N3.3\_O1L46'

NEMO 3.3 ORCA1 L46

**NEMO\_3\_6\_O1L46** = 'N3.6\_O1L46'

NEMO 3.6 ORCA1 L75

### 6.1.3 earthdiagnostics.cdftools

**class** earthdiagnostics.cdftools.**CDFTools** (*path*='')

Bases: `object`

Class to run CDFTools executables

**Parameters** *path* (*str*) – path to CDFTOOLS binaries

**run** (*command*, *input*, *output*=None, *options*=None, *log\_level*=20)

Runs one of the CDFTools

#### Parameters

- **command** (*str*) – executable to run
- **input** (*str*) – input file
- **output** – output file. Not all tools support this parameter
- **options** (*str* / *list*) – options for the tool.
- **log\_level** (*int*) – log level at which the output of the cdftool command will be added

### 6.1.4 earthdiagnostics.datamanager

**class** earthdiagnostics.datamanager.**DataManager** (*exp\_manager*, *institution*, *model*, *expid*,  
*datafolder*, *frequency*, *experiment\_name*,  
*scratch\_dir*, *nfrp*, *calendar*='standard')

Bases: `object`

Class to manage the data repositories

#### Parameters

- **exp\_manager** (`ExperimentManager`) –
- **institution** (*str*) – CMOR's institution name
- **model** (*str*) – CMOR's model name
- **expid** (*str*) – experiment identifier
- **datafolder** (*str*) – Folder containing the data

- **frequency** (*str*) – default frequency
- **experiment\_name** (*str*) – CMOR’s experiment name
- **scratch\_dir** (*str*) – path to scratch folder
- **nfrp** (*int*) – sample frequency
- **calendar** (*str*) – experiment’s calendar

**static domain\_abbreviation** (*domain*, *frequency*)

Returns the table name for a domain-frequency pair :param domain: variable’s domain :type domain: str  
:param frequency: variable’s frequency :type frequency: str :return: variable’s table name :rtype: str

**extract\_variable** (*file\_path*, *handler*, *frequency*, *member*, *startdate*, *temp*, *variable*)

Extracts a variable from a file and creates the CMOR file

#### Parameters

- **file\_path** (*str*) – path to the file
- **handler** (*netCDF\$.Dataset*) – netCDF4 handler for the file
- **frequency** (*str*) – variable’s frequency
- **member** (*int*) – member
- **startdate** (*str*) – startdate
- **temp** (*str*) – temporal file to use
- **variable** (*str*) – variable’s name

**get\_file** (*domain*, *var*, *startdate*, *member*, *chunk*, *grid=None*, *box=None*, *frequency=None*)

Copies a given file from the CMOR repository to the scratch folder and returns the path to the scratch’s copy

#### Parameters

- **domain** (*str*) – CMOR domain
- **var** (*str*) – variable name
- **startdate** (*str*) – file’s startdate
- **member** (*int*) – file’s member
- **chunk** (*int*) – file’s chunk
- **grid** (*str*) – file’s grid (only needed if it is not the original)
- **box** (*Box*) – file’s box (only needed to retrieve sections or averages)
- **frequency** (*str*) – file’s frequency (only needed if it is different from the default)

**Returns** path to the copy created on the scratch folder

**Return type** *str*

**get\_year** (*domain*, *var*, *startdate*, *member*, *year*, *grid=None*, *box=None*)

Gets all the data corresponding to a given year from the CMOR repository to the scratch folder as one file and returns the path to the scratch’s copy.

#### Parameters

- **year** (*int*) – year to retrieve
- **domain** (*str*) – CMOR domain

- **var** (*str*) – variable name
- **startdate** (*str*) – file's startdate
- **member** (*int*) – file's member
- **grid** (*str*) – file's grid (only needed if it is not the original)
- **box** (*Box*) – file's box (only needed to retrieve sections or averages)

**Returns** path to the copy created on the scratch folder

**Return type** *str*

**prepare\_CMOR\_files** (*force\_rebuild, ocean, atmosphere*)

Prepares the data to be used by the diagnostic.

If CMOR data is not created, it show a warning and closes. In the future, an automatic cmorization procedure will be launched

If CMOR data is available but packed, the procedure will unpack it.

#### Parameters

- **atmosphere** (*bool*) – activates atmosphere files cmorization
- **ocean** (*bool*) – activates ocean files cmorization
- **force\_rebuild** (*bool*) – if True, forces the creation of the CMOR files

#### Returns

**send\_file** (*filetosend, domain, var, startdate, member, chunk=None, grid=None, region=None, box=None, rename\_var=None, frequency=None, year=None, date\_str=None*)

Copies a given file to the CMOR repository. It also automatically converts to netCDF 4 if needed and can merge with already existing ones as needed

#### Parameters

- **date\_str** –
- **year** (*int*) – if frequency is yearly, this parameter is used to give the corresponding year
- **rename\_var** (*str*) – if exists, the given variable will be renamed to the one given by var
- **filetosend** (*str*) – path to the file to send to the CMOR repository
- **region** (*str*) – specifies the region represented by the file. If it is defined, the data will be appended to the CMOR repository as a new region in the file or will overwrite if region was already present
- **domain** (*str*) – CMOR domain
- **var** (*str*) – variable name
- **startdate** (*str*) – file's startdate
- **member** (*int*) – file's member
- **chunk** (*int*) – file's chunk
- **grid** (*str*) – file's grid (only needed if it is not the original)
- **box** (*Box*) – file's box (only needed to retrieve sections or averages)
- **frequency** (*str*) – file's frequency (only needed if it is different from the default)

**Returns** path to the copy created on the scratch folder

**Return type** `str`

**class** `earthdiagnostics.datamanager.UnitConversion` (*source, destiny, factor, offset*)

Bases: `object`

Class to manage unit conversions

**classmethod** `add_conversion` (*conversion*)

Adds a conversion to the dictionary

**Parameters** `conversion` (`UnitConversion`) – conversion to add

**classmethod** `get_conversion_factor_offset` (*input\_units, output\_units*)

Gets the conversion factor and offset for two units . The conversion has to be done in the following way:  
converted = original \* factor + offset

**Parameters**

- **input\_units** (*str*) – original units
- **output\_units** (*str*) – destiny units

**Returns** factor and offset

**Return type** [float, float]

**classmethod** `load_conversions` ()

Load conversions from the configuration file

**class** `earthdiagnostics.datamanager.Variable` (*line*)

Bases: `object`

Class to characterize a CMOR variable. It also contains the static method to make the match between thje original name and the standard name. Requires cmor\_table.csv to work.

**classmethod** `get_variable` (*original\_name*)

Returns the cmor variable instance given a variable name

**Parameters** `original_name` (*str*) – original variable's name

**Returns** CMOR variable

**Return type** `Variable`

**classmethod** `load_variables` ()

Loads the cmor\_table.csv and creates the variables dictionary

### 6.1.5 earthdiagnostics.diagnostic

**class** `earthdiagnostics.diagnostic.Diagnostic` (*data\_manager*)

Bases: `object`

Base class for the diagnostics. Provides a common interface for them and also has a mechanism that allows diagnostic retrieval by name.

**Parameters** `data_manager` (`DataManager`) – data manager that will be used to store and retrieve the necessary data

**alias** = None

Alias to call the diagnostic. Must be overridden at the derived classes

**compute** ()

Calculates the diagnostic and stores the output

Must be implemented by derived classes

**classmethod** `generate_jobs` (*diags, options*)

Generate the instances of the diagnostics that will be run by the manager

Must be implemented by derived classes.

**Parameters**

- **diags** (*Diags*) – diagnostics manager
- **options** (*list[str]*) – list of strings containing the options passed to the diagnostic

**Returns**

**static** `get_diagnostic` (*name*)

Return the class for a diagnostic given its name

**Parameters** **name** (*str*) – diagnostic alias

**Returns** the selected Diagnostic class, None if name can not be found

**Return type** *Diagnostic*

**static** `register` ()

Register a new diagnostic using the given alias. It must be call using the derived class. :param cls: diagnostic class to register :type cls: Diagnostic

## 6.1.6 earthdiagnostics.diags

**class** `earthdiagnostics.diags.Diags` (*config\_file*)

Bases: *object*

Launcher class for the diagnostics

**Parameters** **config\_file** (*str*) – path to the configuration file

**run** ()

Run the diagnostics

`earthdiagnostics.diags.main` ()

Entry point for the Earth Diagnostics. For more detailed documentation, use -h option

## 6.1.7 earthdiagnostics.experimentmanager

**class** `earthdiagnostics.experimentmanager.ExperimentManager` (*startdates, members, num\_chunks, chunk\_size, member\_digits, calendar='standard'*)

Bases: *object*

Encapsulates all chunk related tasks

**Parameters**

- **startdates** (*list[str]*) – startdates to run
- **members** (*list[int]*) – members to run
- **num\_chunks** (*int*) – chunks to run
- **chunk\_size** (*int*) – length of the chunks in months
- **member\_digits** (*int*) – minimum number of digits to use in the member name



- **calendar** (*str*) – experiment’s calendar

**get\_chunk\_list** ()  
Return a list with all the chunks :return: List containing tuples of startdate, member and chunk :rtype: tuple[str, int, int]

**get\_full\_years** (*startdate*)  
Returns the list of full years that are in the given startdate :param startdate: startdate to use :type startdate: str :return: list of full years :rtype: list[int]

**get\_member\_list** ()  
Return a list with all the members :return: List containing tuples of startdate and member :rtype: tuple[str, int, int]

**get\_member\_str** (*member*)  
Returns the member name for a given member number. :param member: member’s number :type member: int :return: member’s name :rtype: str

**get\_year\_chunks** (*startdate*, *year*)  
Get the list of chunks containing timesteps from the given year :param startdate: startdate to use :type startdate: str :param year: reference year :type year: int :return: list of chunks containing data from the given year :rtype: list[int]

### 6.1.8 earthdiagnostics.utils

**class** earthdiagnostics.utils.**TempFile**

Bases: `object`

Class to manage temporal files

**autoclean** = **True**

If True, new temporary files are added to the list for future cleaning

**static clean** ()

Removes all temporary files created with Tempfile until now

**files** = []

List of files to clean automatically

**static get** (*filename=None*, *clean=None*, *suffix='.nc'*)

Gets a new temporal filename, storing it for automated cleaning

**Parameters**

- **suffix** –
- **filename** (*str*) – if it is not none, the function will use this filename instead of a random one
- **clean** (*bool*) – if true, stores filename for cleaning

**Returns** path to the temporal file

**Return type** `str`

**prefix** = 'temp'

Prefix for temporary filenames

**scratch\_folder** = ''

Scratch folder to create temporary files on it

**class** `earthdiagnostics.utils.Utils`

Bases: `object`

Container class for miscellaneous utility methods

**static** `available_cpu_count()`

Number of available virtual or physical CPUs on this systemx

**cdo** = `<cdo.Cdo object>`

An instance of Cdo class ready to be used

**static** `convert2netcdf4(filetoconvert)`

Checks if a file is in netCDF4 format and converts to netCDF4 if not

**Parameters** `filetoconvert` (`str`) – file to convert

**static** `copy_variable(source, destiny, variable, must_exist=True)`

Copies the given variable from source to destiny

**Parameters**

- **source** (`netCDF4.Dataset`) – origin file
- **destiny** (`netCDF4.Dataset`) – destiny file
- **variable** (`str`) – variable to copy
- **must\_exist** (`bool`) – if false, does not raise an error if variable does not exist

**Returns**

**static** `execute_shell_command(command, log_level=10)`

Executes a shell command :param command: command to execute

Log.info('Detailed time for diagnostic class') :param log\_level: log level to use for command output :type

log\_level: int :return: command output :rtype: list

**static** `get_datetime_from_netcdf(handler, time_variable='time')`

Gets a datetime array from a netCDF file

**Parameters**

- **handler** (`netCDF4.Dataset`) – file to read
- **time\_variable** (`str`) – variable to read, by default 'time'

**Returns** Datetime numpy array created from the values stored at the netCDF file

**Return type** `np.array`

**static** `get_file_hash(filepath)`

Returns the MD5 hash for the given filepath :param filepath: path to the file to compute hash on :type

filepath:str :return: file's MD5 hash :rtype: str

**static** `get_mask(basin)`

Returns a numpy array containing the mask for the given basin

**Parameters** `basin` (`Basin`) – basin to retrieve

**Returns** mask

**Return type** `numpy.array`

**static** `move_file(source, destiny)`

Moves a file from source to destiny, creating dirs if necessary

**Parameters**

- **source** (*str*) – path to source
- **destiny** (*str*) – path to destiny

**nco** = <nco.nco.Nco object>

An instance of Nco class ready to be used

**static openCdf** (*filepath*, *mode*='a')

Opens a netCDF file and returns a handler to it

**Parameters**

- **filepath** (*str*) – path to the file
- **mode** (*str*) – mode to open the file. By default, a (append)

**Returns** handler to the file

**Return type** netCDF4.Dataset

**static rename\_variable** (*filepath*, *old\_name*, *new\_name*, *must\_exist*=True, *rename\_dimension*=False)

Rename multiple variables from a NetCDF file :param filepath: path to file :type filepath: str :param old\_name: variable's name to change :type old\_name: str :param new\_name: new name :type new\_name: str :param must\_exist: if True, the function will raise an exception if the variable name does not exist :type must\_exist: bool :param rename\_dimension: if True, also rename dimensions with the same name :type rename\_dimension: bool

**static rename\_variables** (*filepath*, *dic\_names*, *must\_exist*=True, *rename\_dimension*=False)

Rename multiple variables from a NetCDF file :param filepath: path to file :type filepath: str :param dic\_names: dictionary containing old names as keys and new names as values :type dic\_names: dict :param must\_exist: if True, the function will raise an exception if the variable name does not exist :type must\_exist: bool :param rename\_dimension: if True, also rename dimensions with the same name :type rename\_dimension: bool

**static setminmax** (*filename*, *variable\_list*)

Sets the valid\_max and valid\_min values to the current max and min values on the file :param filename: path to file :type filename: str :param variable\_list: list of variables in which valid\_min and valid\_max will be set :type variable\_list: str | list

## 6.2 earthdiagnostics.ocean

### 6.2.1 earthdiagnostics.ocean.areamoc

**class** earthdiagnostics.ocean.areamoc.**AreaMoc** (*data\_manager*, *startdate*, *member*, *chunk*, *basin*, *box*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Compute an Atlantic MOC index by averaging the meridional overturning in a latitude band between 1km and 2km or any other index averaging the meridional overturning in a given basin and a given domain

**Original author** Virginie Guemas <virginie.guemas@bsc.es>

**Contributor** Javier Vegas-Regidor<javier.vegas@bsc.es>

**Created** March 2012

**Last modified** June 2016

**Parameters**

- **data\_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk’s number
- **basin** (*Basin*) – basin to compute
- **box** (*Box*) – box to compute

**alias** = ‘mocarea’

Diagnostic alias for the configuration file

**compute** ()

Runs the diagnostic

**classmethod generate\_jobs** (*diags, options*)

Creates a job for each chunk to compute the diagnostic

#### Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – minimum latitude, maximum latitude, minimum depth, maximum depth, basin=Global

#### Returns

## 6.2.2 earthdiagnostics.ocean.averagesection

**class** earthdiagnostics.ocean.averagesection.**AverageSection** (*data\_manager, startdate, member, chunk, variable, domain, box*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Compute an average of a given zone. The variable MUST be in a regular grid

**Original author** Virginie Guemas <virginie.guemas@bsc.es>

**Contributor** Javier Vegas-Regidor<javier.vegas@bsc.es>

**Created** March 2012

**Last modified** June 2016

#### Parameters

- **data\_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk’s number
- **variable** (*str*) – variable’s name
- **domain** (*str*) – variable’s domain
- **box** (*Box*) – box to use for the average

**alias** = ‘avgsection’

Diagnostic alias for the configuration file

**compute()**

Runs the diagnostic

**classmethod generate\_jobs** (*diags, options*)

Creates a job for each chunk to compute the diagnostic

#### Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – variable, minimum longitude, maximum longitude, minimum latitude, maximum latitude, domain=ocean

#### Returns

### 6.2.3 earthdiagnostics.ocean.convectionsites

**class** earthdiagnostics.ocean.convectionsites.**ConvectionSites** (*data\_manager, start-date, member, chunk, model\_version*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Compute the intensity of convection in the four main convection sites

**Original author** Virginie Guemas <virginie.guemas@bsc.es>

**Contributor** Javier Vegas-Regidor<javier.vegas@bsc.es>

**Created** October 2013

**Last modified** June 2016

#### Parameters

- **data\_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **model\_version** (*str*) – model version

**alias = 'convection'**

Diagnostic alias for the configuration file

**compute()**

Runs the diagnostic

**classmethod generate\_jobs** (*diags, options*)

Creates a job for each chunk to compute the diagnostic

#### Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – None

#### Returns

## 6.2.4 earthdiagnostics.ocean.cutsection

**class** earthdiagnostics.ocean.cutsection.**CutSection** (*data\_manager*, *startdate*, *member*,  
*chunk*, *variable*, *domain*, *zonal*,  
*value*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Cuts a meridional or zonal section

**Original author** Virginie Guemas <virginie.guemas@bsc.es>

**Contributor** Javier Vegas-Regidor<javier.vegas@bsc.es>

**Created** September 2012

**Last modified** June 2016

### Parameters

- **data\_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk’s number
- **variable** (*str*) – variable’s name
- **domain** (*str*) – variable’s domain
- **zonal** (*bool*) – specifies if section is zonal or meridional
- **value** (*int*) – value of the section’s coordinate

**alias** = ‘cutsection’

Diagnostics alias for the configuration file

**compute** ()

Runs the diagnostic

**classmethod generate\_jobs** (*diags*, *options*)

Creates a job for each chunk to compute the diagnostic

### Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – variable, zonal, value, domain=ocean

### Returns

## 6.2.5 earthdiagnostics.ocean.gyres

**class** earthdiagnostics.ocean.gyres.**Gyres** (*data\_manager*, *startdate*, *member*, *chunk*,  
*model\_version*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Compute the intensity of the subtropical and subpolar gyres

**Original author** Virginie Guemas <virginie.guemas@bsc.es>

**Contributor** Javier Vegas-Regidor<javier.vegas@bsc.es>

**Created** October 2013

**Last modified** June 2016

**Parameters**

- **data\_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk’s number
- **model\_version** (*str*) – model version

**alias** = ‘gyres’

Dagnostic alias for the configuration file

**compute** ()

Runs the diagnostic

**classmethod generate\_jobs** (*diags, options*)

Creates a job for each chunk to compute the diagnostic

**Parameters**

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – None

**Returns**

## 6.2.6 earthdiagnostics.ocean.interpolate

**class** earthdiagnostics.ocean.interpolate.**Interpolate** (*data\_manager, startdate, member, chunk, variable, domain, model\_version*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

3-dimensional conservative interpolation to the regular atmospheric grid. It can also be used for 2D (i,j) variables

**Original author** Virginie Guemas <virginie.guemas@bsc.es>

**Contributor** Javier Vegas-Regidor <javier.vegas@bsc.es>

**Created** November 2012

**Last modified** June 2016

**Parameters**

- **data\_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk’s number
- **variable** (*str*) – variable’s name
- **domain** (*str*) – variable’s domain
- **model\_version** (*str*) – model version

**alias** = ‘interp’

Dagnostic alias for the configuration file

**compute** ()

Runs the diagnostic

**classmethod** `generate_jobs` (*diags, options*)

Creates a job for each chunk to compute the diagnostic

**Parameters**

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – variable, domain=ocean

**Returns**

## 6.2.7 earthdiagnostics.ocean.maxmoc

**class** `earthdiagnostics.ocean.maxmoc.MaxMoc` (*data\_manager, startdate, member, year, basin, box*)

Bases: `earthdiagnostics.diagnostic.Diagnostic`

Compute an Atlantic MOC index by finding the maximum of the annual mean meridional overturning in a latitude / depth region

**Original author** Virginie Guemas <virginie.guemas@bsc.es>

**Contributor** Javier Vegas-Regidor<javier.vegas@bsc.es>

**Created** March 2012

**Last modified** June 2016

**Parameters**

- **data\_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **year** (*int*) – year to compute
- **basin** (*Basin*) – basin to compute
- **box** (*Box*) – box to compute

**alias** = 'mocmax'

Diagnostic alias for the configuration file

**compute** ()

Runs the diagnostic

**classmethod** `generate_jobs` (*diags, options*)

Creates a job for each complete year to compute the diagnostic

**Parameters**

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – minimum latitude, maximum latitude, minimum depth, maximum depth, basin=global

**Returns**



## 6.2.8 earthdiagnostics.ocean.mixedlayersaltcontent

**class** `earthdiagnostics.ocean.mixedlayersaltcontent.MixedLayerSaltContent` (*data\_manager*, *start-date*, *member*, *chunk*)

Bases: `earthdiagnostics.diagnostic.Diagnostic`

Compute mixed layer salt content

**Original author** Virginie Guemas <virginie.guemas@bsc.es>

**Contributor** Javier Vegas-Regidor<javier.vegas@bsc.es>

**Created** February 2012

**Last modified** June 2016

### Parameters

- **data\_manager** (`DataManager`) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk’s number

**alias** = ‘mlotstsc’

Diagnostics alias for the configuration file

**compute** ()

Runs the diagnostic

**classmethod generate\_jobs** (*diags*, *options*)

Creates a job for each chunk to compute the diagnostic

### Parameters

- **diags** (`Diags`) – Diagnostics manager class
- **options** (*list[str]*) – None

### Returns

## 6.2.9 earthdiagnostics.ocean.moc

**class** `earthdiagnostics.ocean.moc.Moc` (*data\_manager*, *startdate*, *member*, *chunk*)

Bases: `earthdiagnostics.diagnostic.Diagnostic`

Compute the MOC for oceanic basins

**Original author** Virginie Guemas <virginie.guemas@bsc.es>

**Contributor** Javier Vegas-Regidor<javier.vegas@bsc.es>

**Created** March 2012

**Last modified** June 2016

### Parameters

- **data\_manager** (`DataManager`) – data management object

- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk’s number

**alias** = ‘moc’

Diagnostic alias for the configuration file

**compute** ()

Runs the diagnostic

**classmethod generate\_jobs** (*diags, options*)

Creates a job for each chunk to compute the diagnostic

#### Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – None

#### Returns

## 6.2.10 earthdiagnostics.ocean.psi

**class** earthdiagnostics.ocean.psi.**Psi** (*data\_manager, startdate, member, chunk*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Compute the barotropic stream function

**Original author** Virginie Guemas <virginie.guemas@bsc.es>

**Contributor** Javier Vegas-Regidor<javier.vegas@bsc.es>

**Created** March 2012

**Last modified** June 2016

#### Parameters

- **data\_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk’s number

**alias** = ‘psi’

Diagnostic alias for the configuration file

**compute** ()

Runs the diagnostic

**classmethod generate\_jobs** (*diags, options*)

Creates a job for each chunk to compute the diagnostic

#### Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – None

#### Returns

### 6.2.11 earthdiagnostics.ocean.siasiesiv

**class** earthdiagnostics.ocean.siasiesiv.**Siasiesiv** (*data\_manager, basin, startdate, member, chunk, mask*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Compute the sea ice extent , area and volume in both hemispheres or a specified region.

**Original author** Virginie Guemas <virginie.guemas@bsc.es>

**Contributor** Neven Fuckar <neven.fuckar@bsc.es>

**Contributor** Ruben Cruz <ruben.cruzgarcia@bsc.es>

**Contributor** Javier Vegas-Regidor <javier.vegas@bsc.es>

**Created** April 2012

**Last modified** June 2016

**alias** = 'siasiesiv'

Diagnostic alias for the configuration file

**compute** ()

Runs the diagnostic

**classmethod generate\_jobs** (*diags, options*)

Creates a job for each chunk to compute the diagnostic

#### Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – basin

#### Returns

### 6.2.12 earthdiagnostics.ocean.verticalmean

**class** earthdiagnostics.ocean.verticalmean.**VerticalMean** (*data\_manager, startdate, member, chunk, variable, box*)

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Chooses vertical level in ocean, or vertically averages between 2 or more ocean levels

**Original author** Virginie Guemas <virginie.guemas@bsc.es>

**Contributor** Eleftheria Exarchou <eleftheria.exarchou@bsc.es>

**Contributor** Javier Vegas-Regidor <javier.vegas@bsc.es>

**Created** February 2012

**Last modified** June 2016

#### Parameters

- **data\_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **variable** (*str*) – variable to average

- **box** (*Box*) – box used to restrict the vertical mean

**alias** = 'vertmean'

Diagnostic alias for the configuration file

**compute** ()

Runs the diagnostic

**classmethod generate\_jobs** (*diags, options*)

Creates a job for each chunk to compute the diagnostic

#### Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – variable, minimum depth (level), maximum depth (level)

#### Returns

### 6.2.13 earthdiagnostics.ocean.verticalmeanmeters

```
class earthdiagnostics.ocean.verticalmeanmeters.VerticalMeanMeters (data_manager,  
startdate,  
member,  
chunk, vari-  
able, box)
```

Bases: *earthdiagnostics.diagnostic.Diagnostic*

Averages vertically any given variable

**Original author** Virginie Guemas <virginie.guemas@bsc.es>

**Contributor** Javier Vegas-Regidor<javier.vegas@bsc.es>

**Created** February 2012

**Last modified** June 2016

#### Parameters

- **data\_manager** (*DataManager*) – data management object
- **startdate** (*str*) – startdate
- **member** (*int*) – member number
- **chunk** (*int*) – chunk's number
- **variable** (*str*) – variable to average
- **box** (*Box*) – box used to restrict the vertical mean

**alias** = 'vertmeanmeters'

Diagnostic alias for the configuration file

**compute** ()

Runs the diagnostic

**classmethod generate\_jobs** (*diags, options*)

Creates a job for each chunk to compute the diagnostic

#### Parameters

- **diags** (*Diags*) – Diagnostics manager class
- **options** (*list[str]*) – variable, minimum depth (meters), maximum depth (meters)

## Returns



**e**

- `earthdiagnostics.box`, [11](#)
- `earthdiagnostics.cdftools`, [16](#)
- `earthdiagnostics.constants`, [12](#)
- `earthdiagnostics.datamanager`, [16](#)
- `earthdiagnostics.diagnostic`, [19](#)
- `earthdiagnostics.diags`, [20](#)
- `earthdiagnostics.experimentmanager`, [20](#)
- `earthdiagnostics.ocean.areamoc`, [23](#)
- `earthdiagnostics.ocean.averagesection`,  
[24](#)
- `earthdiagnostics.ocean.convectionsites`,  
[25](#)
- `earthdiagnostics.ocean.cutsection`, [26](#)
- `earthdiagnostics.ocean.gyres`, [26](#)
- `earthdiagnostics.ocean.interpolate`, [27](#)
- `earthdiagnostics.ocean.maxmoc`, [28](#)
- `earthdiagnostics.ocean.mixedlayersaltcontent`,  
[29](#)
- `earthdiagnostics.ocean.moc`, [29](#)
- `earthdiagnostics.ocean.psi`, [30](#)
- `earthdiagnostics.ocean.siasiesiv`, [31](#)
- `earthdiagnostics.ocean.verticalmean`, [31](#)
- `earthdiagnostics.ocean.verticalmeanmeters`,  
[32](#)
- `earthdiagnostics.utils`, [21](#)





## A

`add_conversion()` (earthdiagnostics.datamanager.UnitConversion class method), 19

`alias` (earthdiagnostics.diagnostic.Diagnostic attribute), 19

`alias` (earthdiagnostics.ocean.areamoc.AreaMoc attribute), 24

`alias` (earthdiagnostics.ocean.averagesection.AverageSection attribute), 24

`alias` (earthdiagnostics.ocean.convectionsites.ConvectionSites attribute), 25

`alias` (earthdiagnostics.ocean.cutsection.CutSection attribute), 26

`alias` (earthdiagnostics.ocean.gyres.Gyres attribute), 27

`alias` (earthdiagnostics.ocean.interpolate.Interpolate attribute), 27

`alias` (earthdiagnostics.ocean.maxmoc.MaxMoc attribute), 28

`alias` (earthdiagnostics.ocean.mixedlayersaltcontent.MixedLayerSaltContent attribute), 29

`alias` (earthdiagnostics.ocean.moc.Moc attribute), 30

`alias` (earthdiagnostics.ocean.psi.Psi attribute), 30

`alias` (earthdiagnostics.ocean.siasiesiv.Siasiesiv attribute), 31

`alias` (earthdiagnostics.ocean.verticalmean.VerticalMean attribute), 32

`alias` (earthdiagnostics.ocean.verticalmeanmeters.VerticalMeanMeters attribute), 32

`Antarctic` (earthdiagnostics.constants.Basins attribute), 12

`AntarcticAtlantic` (earthdiagnostics.constants.Basins attribute), 12

`AntarcticIndian` (earthdiagnostics.constants.Basins attribute), 12

`Arctic` (earthdiagnostics.constants.Basins attribute), 12

`ArcticMarginalSeas` (earthdiagnostics.constants.Basins attribute), 12

`ArcticNorthAtlantic` (earthdiagnostics.constants.Basins attribute), 12

`AreaMoc` (class in earthdiagnostics.ocean.areamoc), 23

`Atlantic` (earthdiagnostics.constants.Basins attribute), 12

`autoclean` (earthdiagnostics.utils.TempFile attribute), 21

`available_cpu_count()` (earthdiagnostics.utils.Utils static method), 22

`AverageSection` (class in earthdiagnostics.ocean.averagesection), 24

## B

`Baffin` (earthdiagnostics.constants.Basins attribute), 12

`Baffin_Bay` (earthdiagnostics.constants.Basins attribute), 12

`Baltic_Sea` (earthdiagnostics.constants.Basins attribute), 12

`Barents_Sea` (earthdiagnostics.constants.Basins attribute), 12

`BarKara` (earthdiagnostics.constants.Basins attribute), 12

`Basin` (class in earthdiagnostics.constants), 12

`Basins` (class in earthdiagnostics.constants), 12

`Beaufort_Chukchi_Sea` (earthdiagnostics.constants.Basins attribute), 13

`Beaufort_Sea` (earthdiagnostics.constants.Basins attribute), 13

`Bellingshausen_Sea` (earthdiagnostics.constants.Basins attribute), 13

`Bering` (earthdiagnostics.constants.Basins attribute), 13

`Bering_Strait` (earthdiagnostics.constants.Basins attribute), 13

`Box` (class in earthdiagnostics.box), 11

`box` (earthdiagnostics.constants.Basin attribute), 12

## C

`Canadian_Waters` (earthdiagnostics.constants.Basins attribute), 13

`CanArch` (earthdiagnostics.constants.Basins attribute), 13

`Caspian_Sea` (earthdiagnostics.constants.Basins attribute), 13

`CDFTools` (class in earthdiagnostics.cdftools), 16

`cdo` (earthdiagnostics.utils.Utils attribute), 22

`Central_Arctic` (earthdiagnostics.constants.Basins attribute), 13

`Chukchi_Sea` (earthdiagnostics.constants.Basins attribute), 13

`clean()` (earthdiagnostics.utils.TempFile static method), 21

compute() (earthdiagnostics.diagnostic.Diagnostic method), 19

compute() (earthdiagnostics.ocean.areamoc.AreaMoc method), 24

compute() (earthdiagnostics.ocean.averagesection.AverageSection method), 24

compute() (earthdiagnostics.ocean.convectionsites.ConvectionSites method), 25

compute() (earthdiagnostics.ocean.cutsection.CutSection method), 26

compute() (earthdiagnostics.ocean.gyres.Gyres method), 27

compute() (earthdiagnostics.ocean.interpolate.Interpolate method), 27

compute() (earthdiagnostics.ocean.maxmoc.MaxMoc method), 28

compute() (earthdiagnostics.ocean.mixedlayersaltcontent.MixedLayerSaltContent method), 29

compute() (earthdiagnostics.ocean.moc.Moc method), 30

compute() (earthdiagnostics.ocean.psi.Psi method), 30

compute() (earthdiagnostics.ocean.siasiesiv.Siasiesiv method), 31

compute() (earthdiagnostics.ocean.verticalmean.VerticalMean method), 32

compute() (earthdiagnostics.ocean.verticalmeanmeters.VerticalMeanMeters method), 32

ConvectionSites (class in earthdiagnostics.ocean.convectionsites), 25

convert2netcdf4() (earthdiagnostics.utils.Utils static method), 22

copy\_variable() (earthdiagnostics.utils.Utils static method), 22

CutSection (class in earthdiagnostics.ocean.cutsection), 26

earthdiagnostics.diagnostic (module), 19

earthdiagnostics.diagnostics (module), 20

earthdiagnostics.experimentmanager (module), 20

earthdiagnostics.ocean.areamoc (module), 23

earthdiagnostics.ocean.averagesection (module), 24

earthdiagnostics.ocean.convectionsites (module), 25

earthdiagnostics.ocean.cutsection (module), 26

earthdiagnostics.ocean.gyres (module), 26

earthdiagnostics.ocean.interpolate (module), 27

earthdiagnostics.ocean.maxmoc (module), 28

earthdiagnostics.ocean.mixedlayersaltcontent (module), 29

earthdiagnostics.ocean.moc (module), 29

earthdiagnostics.ocean.psi (module), 30

earthdiagnostics.ocean.siasiesiv (module), 31

earthdiagnostics.ocean.verticalmean (module), 31

earthdiagnostics.ocean.verticalmeanmeters (module), 32

earthdiagnostics.utils (module), 21

East\_Siberian\_Sea (earthdiagnostics.constants.Basins attribute), 13

Eastern\_Central\_Arctic (earthdiagnostics.constants.Basins attribute), 13

ECEARTH\_2\_3\_O1L42 (earthdiagnostics.constants.Models attribute), 15

ECEARTH\_3\_0\_O1L46 (earthdiagnostics.constants.Models attribute), 15

ECEARTH\_3\_0\_O25L46 (earthdiagnostics.constants.Models attribute), 15

ECEARTH\_3\_0\_O25L75 (earthdiagnostics.constants.Models attribute), 15

ECEARTH\_3\_1\_O25L75 (earthdiagnostics.constants.Models attribute), 15

execute\_shell\_command() (earthdiagnostics.utils.Utils static method), 22

ExperimentManager (class in earthdiagnostics.experimentmanager), 20

extract\_variable() (earthdiagnostics.datamanager.DataManager method), 17

## D

DataManager (class in earthdiagnostics.datamanager), 16

depth\_in\_meters (earthdiagnostics.box.Box attribute), 11

Diagnostic (class in earthdiagnostics.diagnostic), 19

Diagnostics (class in earthdiagnostics.diagnostics), 20

domain\_abbreviation() (earthdiagnostics.datamanager.DataManager static method), 17

## E

earthdiagnostics.box (module), 11

earthdiagnostics.cdftools (module), 16

earthdiagnostics.constants (module), 12

earthdiagnostics.datamanager (module), 16

## F

files (earthdiagnostics.utils.TempFile attribute), 21

Fram\_Strait (earthdiagnostics.constants.Basins attribute), 13

fullname (earthdiagnostics.constants.Basin attribute), 12

## G

generate\_jobs() (earthdiagnostics.diagnostic.Diagnostic class method), 20

generate\_jobs() (earthdiagnostics.ocean.areamoc.AreaMoc class method), 24

generate\_jobs() (earthdiagnostics.ocean.averagesection.AverageSection

- class method), 25
- generate\_jobs() (earthdiagnostics.ocean.convectionsites.ConvectionSites class method), 25
- generate\_jobs() (earthdiagnostics.ocean.cutsection.CutSection class method), 26
- generate\_jobs() (earthdiagnostics.ocean.gyres.Gyres class method), 27
- generate\_jobs() (earthdiagnostics.ocean.interpolate.Interpolate class method), 27
- generate\_jobs() (earthdiagnostics.ocean.maxmoc.MaxMoc class method), 28
- generate\_jobs() (earthdiagnostics.ocean.mixedlayersaltcontent.MixedLayerSaltContent class method), 29
- generate\_jobs() (earthdiagnostics.ocean.moc.Moc class method), 30
- generate\_jobs() (earthdiagnostics.ocean.psi.Psi class method), 30
- generate\_jobs() (earthdiagnostics.ocean.siasiesiv.Siasiesiv class method), 31
- generate\_jobs() (earthdiagnostics.ocean.verticalmean.VerticalMean class method), 32
- generate\_jobs() (earthdiagnostics.ocean.verticalmeanmeters.VerticalMeanMeters class method), 32
- get() (earthdiagnostics.utils.TempFile static method), 21
- get\_chunk\_list() (earthdiagnostics.experimentmanager.ExperimentManager method), 21
- get\_conversion\_factor\_offset() (earthdiagnostics.datamanager.UnitConversion class method), 19
- get\_datetime\_from\_netcdf() (earthdiagnostics.utils.Utils static method), 22
- get\_depth\_str() (earthdiagnostics.box.Box method), 11
- get\_diagnostic() (earthdiagnostics.diagnostic.Diagnostic static method), 20
- get\_file() (earthdiagnostics.datamanager.DataManager method), 17
- get\_file\_hash() (earthdiagnostics.utils.Utils static method), 22
- get\_full\_years() (earthdiagnostics.experimentmanager.ExperimentManager method), 21
- get\_lat\_str() (earthdiagnostics.box.Box method), 11
- get\_lon\_str() (earthdiagnostics.box.Box method), 11
- get\_mask() (earthdiagnostics.utils.Utils static method), 22
- get\_member\_list() (earthdiagnostics.experimentmanager.ExperimentManager method), 21
- get\_member\_str() (earthdiagnostics.experimentmanager.ExperimentManager method), 21
- get\_variable() (earthdiagnostics.datamanager.Variable class method), 19
- get\_year() (earthdiagnostics.datamanager.DataManager method), 17
- get\_year\_chunks() (earthdiagnostics.experimentmanager.ExperimentManager method), 21
- Global (earthdiagnostics.constants.Basins attribute), 13
- Global\_Ocean (earthdiagnostics.constants.Basins attribute), 13
- Greenland Sea (earthdiagnostics.constants.Basins attribute), 13
- Grnland (earthdiagnostics.constants.Basins attribute), 13
- Gyres (class in earthdiagnostics.ocean.gyres), 26
- ## H
- Hudson (earthdiagnostics.constants.Basins attribute), 13
- ## I
- Icelandic\_Sea (earthdiagnostics.constants.Basins attribute), 13
- Indian (earthdiagnostics.constants.Basins attribute), 13
- IndoPacific (earthdiagnostics.constants.Basins attribute), 13
- Interpolate (class in earthdiagnostics.ocean.interpolate), 27
- ## K
- Kara\_Gate\_Strait (earthdiagnostics.constants.Basins attribute), 14
- Kara\_Sea (earthdiagnostics.constants.Basins attribute), 14
- ## L
- Labrador\_Sea (earthdiagnostics.constants.Basins attribute), 14
- Laptev\_East\_Siberian\_Chukchi\_Seas (earthdiagnostics.constants.Basins attribute), 14
- Laptev\_East\_Siberian\_Seas (earthdiagnostics.constants.Basins attribute), 14
- Laptev\_Sea (earthdiagnostics.constants.Basins attribute), 14
- Lincoln\_Sea (earthdiagnostics.constants.Basins attribute), 14

`load_conversions()` (earthdiagnostics.datamanager.UnitConversion class method), 19

`load_variables()` (earthdiagnostics.datamanager.Variable class method), 19

## M

`main()` (in module earthdiagnostics.diagnostics), 20

`max_depth` (earthdiagnostics.box.Box attribute), 11

`max_lat` (earthdiagnostics.box.Box attribute), 11

`max_lon` (earthdiagnostics.box.Box attribute), 11

`MaxMoc` (class in earthdiagnostics.ocean.maxmoc), 28

`Mediterranean_Sea` (earthdiagnostics.constants.Basins attribute), 14

`min_depth` (earthdiagnostics.box.Box attribute), 11

`min_lat` (earthdiagnostics.box.Box attribute), 11

`min_lon` (earthdiagnostics.box.Box attribute), 11

`MixedLayerSaltContent` (class in earthdiagnostics.ocean.mixedlayersaltcontent), 29

`Moc` (class in earthdiagnostics.ocean.moc), 29

`Models` (class in earthdiagnostics.constants), 15

`move_file()` (earthdiagnostics.utils.Utils static method), 22

## N

`Nares_Strait` (earthdiagnostics.constants.Basins attribute), 14

`nco` (earthdiagnostics.utils.Utils attribute), 23

`NEMO_3_2_O1L42` (earthdiagnostics.constants.Models attribute), 16

`NEMO_3_3_O1L46` (earthdiagnostics.constants.Models attribute), 16

`NEMO_3_6_O1L46` (earthdiagnostics.constants.Models attribute), 16

`NEMOVAR_O1L42` (earthdiagnostics.constants.Models attribute), 16

`Nordic_Barents_Seas` (earthdiagnostics.constants.Basins attribute), 14

`Nordic_Seas` (earthdiagnostics.constants.Basins attribute), 14

`North_Atlantic_Arctic` (earthdiagnostics.constants.Basins attribute), 14

`North_Hemisphere_Ocean` (earthdiagnostics.constants.Basins attribute), 14

`NorthAtlantic` (earthdiagnostics.constants.Basins attribute), 14

`NorthPacific` (earthdiagnostics.constants.Basins attribute), 14

`NorthWest_Passage` (earthdiagnostics.constants.Basins attribute), 14

`Norwegian_Sea` (earthdiagnostics.constants.Basins attribute), 14

## O

`Okhotsk` (earthdiagnostics.constants.Basins attribute), 14

`openCdf()` (earthdiagnostics.utils.Utils static method), 23

`OpenOcean` (earthdiagnostics.constants.Basins attribute), 14

## P

`Pacific` (earthdiagnostics.constants.Basins attribute), 14

`parse()` (earthdiagnostics.constants.Basins class method), 15

`prefix` (earthdiagnostics.utils.TempFile attribute), 21

`prepare_CMOR_files()` (earthdiagnostics.datamanager.DataManager method), 18

`Psi` (class in earthdiagnostics.ocean.psi), 30

## R

`register()` (earthdiagnostics.diagnostics.Diagnostic static method), 20

`rename_variable()` (earthdiagnostics.utils.Utils static method), 23

`rename_variables()` (earthdiagnostics.utils.Utils static method), 23

`Ross_Sea` (earthdiagnostics.constants.Basins attribute), 14

`run()` (earthdiagnostics.cdftools.CDFTools method), 16

`run()` (earthdiagnostics.diagnostics.Diagnostics method), 20

## S

`scratch_folder` (earthdiagnostics.utils.TempFile attribute), 21

`send_file()` (earthdiagnostics.datamanager.DataManager method), 18

`Serreze_Arctic` (earthdiagnostics.constants.Basins attribute), 15

`setminmax()` (earthdiagnostics.utils.Utils static method), 23

`shortname` (earthdiagnostics.constants.Basins attribute), 12

`Siasiesiv` (class in earthdiagnostics.ocean.siasiesiv), 31

`Southern_Hemisphere` (earthdiagnostics.constants.Basins attribute), 15

`StLawr` (earthdiagnostics.constants.Basins attribute), 15

`Subpolar_Gyre` (earthdiagnostics.constants.Basins attribute), 15

## T

`TempFile` (class in earthdiagnostics.utils), 21

`TotalArc` (earthdiagnostics.constants.Basins attribute), 15

`TropicalAtlantic` (earthdiagnostics.constants.Basins attribute), 15

`TropicalIndian` (earthdiagnostics.constants.Basins attribute), 15

`TropicalPacific` (earthdiagnostics.constants.Basins attribute), 15

## U

UnitConversion (class in earthdiagnostics.datamanager), [19](#)

Utils (class in earthdiagnostics.utils), [21](#)

## V

Variable (class in earthdiagnostics.datamanager), [19](#)

VerticalMean (class in earthdiagnostics.ocean.verticalmean), [31](#)

VerticalMeanMeters (class in earthdiagnostics.ocean.verticalmeanmeters), [32](#)

Vilkitsky\_Strait (earthdiagnostics.constants.Basins attribute), [15](#)

## W

Weddell\_Sea (earthdiagnostics.constants.Basins attribute), [15](#)

Western\_Central\_Arctic (earthdiagnostics.constants.Basins attribute), [15](#)