ASAGI User Manual

0.3

Generated by Doxygen 1.7.6.1

Fri Jan 30 2015 12:52:33

Contents

1 Building and Installing ASAGI

1.1 Pre-requirements

1.1.1 MPI

ASAGI makes use of the RMA (Remote Memory Access) API of the MPI-2 standard to transfer data. An MPI library that supports the new standard is required.

1.1.2 NetCDF

ASAGI uses the NetCDF library (`http://www.unidata.ucar.edu/software/netcdf/`) to load data files.

1.2 Compilation

To generate the Makefiles, CMake is used. For CMake it is recommend to keep source and build directory apart:

```
{.sh}
mkdir build
cd build
cmake <path/to/asagi_sources>
```

Several environment variables affect the behavior of CMake. They must be set before running "cmake".

- **Compiler** The compiler can be selected by setting `CC` (C compiler), `CXX` (C++ compiler) and `FC` (Fortran compiler) environment variables. C and Fortran compiler are only required for C and Fortran examples and tests.
- **Libraries** The `CMAKE_PREFIX_PATH` is used when searching for the MPI, NetCDF and PNG library. If NetCDF was configured with `--prefix=<install-_dir>` for example, set `CMAKE_PREFIX_PATH=<install_dir>`.

Besides the environment variables, you can change the behavior by setting internal CMake variables. They can be configured by adding one ore more `--D<variable>=<value>` options when running "cmake". These variables can also be changed later with the following command:

```
{.sh}
ccmake <path/to/asagi_build>
```

The important variables are listed below. Most of the variables are ASAGI specific and will not work with other CMake projects.

- **CMAKE_BUILD_TYPE = Debug | Release** When set to "Debug", additional runtime checks are enabled as well as debug messages. `[Release]`
- **CMAKE_INSTALL_PREFIX** Installation directory for ASAGI. `[/usr/local/]`
- **EXAMPLES = ON | OFF** Compile example programs. `[OFF]`
- **FORTRAN_SUPPORT = ON | OFF** Compile with Fortran support. `[ON]`
- **SHARED_LIB = ON | OFF** Build shared library. `[ON]`
- **STATIC_LIB = ON | OFF** Build static library. `[OFF]`
- **TESTS = ON | OFF** Compile tests. `[OFF]`
- **THREADSAFETY = ON | OFF** If enabled all ASAGI functions are thread-safe. This is required, for example, if ASAGI is used in hybrid MPI/OpenMP programs. `[ON]`
- **NOMPI = ON | OFF** Do not compile with MPI support. All algorithms that require communication will be disabled. `[OFF]`

1.3 Tests

If you have enabled the tests, you can run them with the following command:

```
{.sh}
make test
```

1.4 Installation

To install ASAGI simply run:

```
{.sh}
make install
```

This will install the (static and/or shared) library as well as the header files.

1.5 Static library

When a program is linked against the static version of ASAGI, make sure to link against netCDF (and PNG if installed) as well.

2 Using ASAGI

2.1 Minimal examples

These are minimal C, C++ and Fortran examples that load a 2-dimensional grid and print the value at (0,0). In each case the grid contains floating point values.

C example:

```
#include <mpi.h>
#include <asagi.h>
#include <stdio.h>

int main(int argc, char** argv)
{
  MPI_Init(&argc, &argv);

  grid_handle* grid = grid_create(GRID_FLOAT, GRID_NO_HINT, 1);

  if (grid_open(grid, "/path/to/netcdf/file.nc", 0) != GRID_SUCCESS) {
    printf("Could not load file\n");
    return 1;
  }

  printf("Value at (0,0): %f\n", grid_get_float_2d(grid, 0, 0, 0));

  grid_close(grid);

  MPI_Finalize();

  return 0;
}
```

C++ example:

```
#include <mpi.h>
#include <asagi.h>
#include <iostream>

using namespace asagi;

int main(int argc, char** argv)
{
  MPI_Init(&argc, &argv);

  Grid* grid = Grid::create();

  if (grid->open("/path/to/netcdf/file.nc") != Grid::SUCCESS) {
    std::cout << "Could not load file" << std::endl;
    return 1;
  }

  std::cout << "Value at (0,0): " << grid->getFloat2D(0, 0) << std::endl;

  // The same as: "Grid::close(grid);"
  delete grid;

  MPI_Finalize();

  return 0;
}
```

Fortran example:

```
! You have two options:
! - Include the module file _once_ in your project:
!include 'asagi.f90'
! - Compile and link the module file as any other file in your project

program minimal
  use mpi
  use asagi
  implicit none

  integer :: grid_id
  integer :: error

  call MPI_Init( error )

  grid_id = grid_create( )

  if( grid_open( grid_id, "/path/to/netcdf/file.nc" ) /= GRID_SUCCESS ) then
    write (*,*) "Could not load file"
    call exit(1)
  end if

  write (*,*) "Value at (0,0):", grid_get_float( grid_id, 0.d+0, 0.d+0 )

  call grid_close( grid_id )

  call MPI_Finalize( error )
end program minimal
```

2.2 Dimensions

ASAGI supports grids with up to three dimensions. The number of dimension cannot be specified by calling an ASAGI function but depends on the NetCDF input file. For example, to access an integer of a 2-dimensional grid in C++, use `getInt2D`. For a 3-dimensional grid, the corresponding call is `getInt3D`.

2.3 Level of detail

A grid can have multiple resolutions. Each resolution is identified by a level id (level of detail). If the number of levels is not specified when creating a grid, the grid will contain only one level of detail. In this case you can also omit the level id in all other functions, since level 0 will be used by default. (C does not support default arguments or overloading, therefore omitting arguments is not possible when using the C interface.)

For grids with multiple levels asagi::Grid::open() must be called once for each level. - Several levels can be stored in a single NetCDF file with different variable names. (Use asagi::Grid::setParam() to specify the variable name.) The coarsest resolution should have the level id 0. With ascending level id, the resolution should get finer. When accessing values with any `get` function, the level of detail can be selected with the last argument. The function asagi::Grid::close() has to be called only once for the whole grid.

2.4 Coordinate mapping

ASAGI distinguishes between actual coordinates and internal array indexes. All functions, that return a grid value, expect actual coordinates. ASAGI maps each coordinate to an array index using the coordinate variables from the NetCDF file (see section NetCDF files on how specify coordinate variables in NetCDF files). If no coordinate variable is available, the mapping is omitted. After the mapping, the coordinate is rounded to the nearest array index. ASAGI does not interpolate between array values.

The actual range of the grid can be obtained with the `getMin`/`getMax` functions. - They also return coordinates, not array indexes. It is erroneous to access values outside range of the grid.

The range of a dimension can be $(-\infty,\infty)$. This is the case if the size of the dimension in the NetCDF file is one.

2.5 Value position

ASAGI supports cell-centered and vertex-centered grids. The value position can be switched with asagi::Grid::setParam().

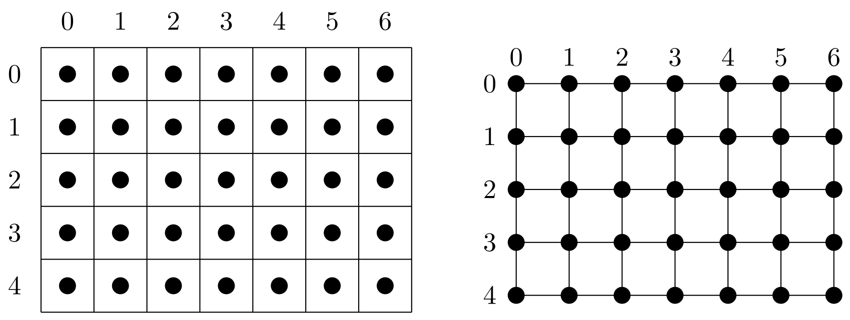

Figure 1: Cell-centered and vertex-centered grids

2.6 NetCDF files

All NetCDF files opened with ASAGI should respect the COARDS conventions (`http-://ferret.wrc.noaa.gov/noaa_coop/coop_cdf_profile.html`). - However, ASAGI has some further limitations:

- The attributes `scale_factor` and `add_offset` are ignored. Besides conversion between data types, ASAGI does not modify the values.
- Since ASAGI does not change the NetCDF file, all values have to be present in the file. Attributes, like `_FillValue` and `missing_value`, are not supported.
- ASAGI is not aware of any units. It is up to the user of the library to interpret the values correctly.

- Variables with more than three dimensions are not supported.

It is possible to open a NetCDF file by different grids or levels at the same time. This allows you, for example, to store all levels of one grid in a single NetCDF file. In this case the levels must be distinguished by the variable names.

2.7 Multi-thread support

When compiled with `THREADSAFTY=ON` (see section Compilation) all functions are thread-safe. However, there are some restrictions due to MPI implementations. To receive values from a grid with different threads MPI must support at least `MPI_THREAD_SERIALIZED`. If you want to `open` or `close` several grids at the same time, support for `MPI_THREAD_MULTIPLE` is required.

3 ASAGI on the LRZ Linux-Cluster

This part describes the installation on the ICE and MPP segments of the LRZ Linux--Cluster. The segments use different MPI versions. Thus, ASAGI compiled on one segment may not work on the other.

3.1 Pre-requirements

3.1.1 Additional modules

Load the CMake module and switch to the latest Intel C compiler:

```
{.sh}
module load cmake
module unload ccomp
module load ccomp/intel/13.1
module unload gcc
module load gcc/4.7
module load netcdf
```

The GCC module is required to get support for additional C++11 template libraries.

3.2 Compilation

Selecting Intel compiler and correct NetCDF and MPI libraries can be achieved with the following command:

```
{.sh}
CXX=icpc CMAKE_PREFIX_PATH="$NETCDF_BASE:$MPI_BASE" \
  cmake <path/to/asagi_sources> \
  -DCMAKE_INSTALL_PREFIX=<install_dir>
```

4 Troubleshooting

4.1 CMake does not find MPI

On some plattforms, CMake has problems finding MPI. Try to set the environment variable `CMAKE_PREFIX_PATH` (see section Building and Installing ASAGI) or select the MPI compiler before running CMake by setting the enviroment variable `CXX`.

4.2 The program hangs

Due to a bug (http://software.intel.com/en-us/forums/showthread.php?t=103456) in the Intel MPI library (version 4.0 update 3 and probably earlier versions) the remote memory access in ASAGI does not work properly. This only happens when fabric is set to "ofa" or "shm:ofa". Selecting a different fabric by changing the environment variable "I_MPI_FABRICS" solves the problem.

4.3 The program fails with "PMPI_Win_create: Assertion 'winptr->lock_table[i]' failed" or "function:MPI_WIN_LOCK, Invalid win argument"

The SGI Message Passing Toolkit uses a special mapped memory for one-sided communication. For large grids the default size of mapped memory may be too small. It is possible to increase the size by setting the environment variable `MPI_MAPPED_HEAP_SIZE`.

5 Module Documentation

5.1 Fortran Interface

Classes

- module asagi

    ASAGI Fortran Interface. More...
- interface asagi::grid_get_byte

    Interface for arbitrary dimensions. More...
- interface asagi::grid_get_int

    Interface for arbitrary dimensions. More...
- interface asagi::grid_get_long

    Interface for arbitrary dimensions. More...
- interface asagi::grid_get_float

    Interface for arbitrary dimensions. More...
- interface asagi::grid_get_double

    Interface for arbitrary dimensions. More...
- interface asagi::grid_get_buf

    Interface for arbitrary dimensions. More...

Functions

- integer(kind=c_int) function asagi::grid_create_c (grid_type, hint, levels)
- integer(kind=c_int) function asagi::grid_set_comm (grid_id, comm)
- real(kind=c_double) function asagi::grid_min_x (grid_id)
- real(kind=c_double) function asagi::grid_max_x (grid_id)
- real(kind=c_double) function asagi::grid_min_y (grid_id)
- real(kind=c_double) function asagi::grid_max_y (grid_id)
- real(kind=c_double) function asagi::grid_min_z (grid_id)
- real(kind=c_double) function asagi::grid_max_z (grid_id)
- real(kind=c_double) function asagi::grid_delta_x (grid_id)
- real(kind=c_double) function asagi::grid_delta_y (grid_id)
- real(kind=c_double) function asagi::grid_delta_z (grid_id)
- integer(kind=c_int) function asagi::grid_var_size (grid_id)
- subroutine asagi::grid_close (grid_id)
- integer function asagi::grid_create (grid_type, hint, levels)
- integer function asagi::grid_create_array (grid_basictype, hint, levels)
- integer function asagi::grid_create_struct (count, block_length, displacments, types, hint, levels)
- integer function asagi::grid_set_param (grid_id, name, value, level)
- integer function asagi::grid_open (grid_id, filename, level)
- character function asagi::grid_get_byte_1d (grid_id, x, level)
- integer function asagi::grid_get_int_1d (grid_id, x, level)
- integer(kind=c_long) function asagi::grid_get_long_1d (grid_id, x, level)
- real function asagi::grid_get_float_1d (grid_id, x, level)
- real(kind=c_double) function asagi::grid_get_double_1d (grid_id, x, level)
- subroutine asagi::grid_get_buf_1d (grid_id, buf, x, level)
- character function asagi::grid_get_byte_2d (grid_id, x, y, level)
- integer function asagi::grid_get_int_2d (grid_id, x, y, level)
- integer(kind=c_long) function asagi::grid_get_long_2d (grid_id, x, y, level)
- real function asagi::grid_get_float_2d (grid_id, x, y, level)
- real(kind=c_double) function asagi::grid_get_double_2d (grid_id, x, y, level)
- subroutine asagi::grid_get_buf_2d (grid_id, buf, x, y, level)
- character function asagi::grid_get_byte_3d (grid_id, x, y, z, level)
- integer function asagi::grid_get_int_3d (grid_id, x, y, z, level)
- integer(kind=c_long) function asagi::grid_get_long_3d (grid_id, x, y, z, level)
- real function asagi::grid_get_float_3d (grid_id, x, y, z, level)
- real(kind=c_double) function asagi::grid_get_double_3d (grid_id, x, y, z, level)
- subroutine asagi::grid_get_buf_3d (grid_id, buf, x, y, z, level)

Variables

- integer, parameter asagi::GRID_NO_HINT = z'0'
- integer, parameter asagi::GRID_HAS_TIME = z'1'
- integer, parameter asagi::GRID_NOMPI = z'2'
- integer, parameter asagi::SMALL_CACHE = z'4'
- integer, parameter asagi::GRID_LARGE_GRID = z'8'
- integer, parameter asagi::GRID_ADAPTIVE = z'10'
- integer, parameter asagi::GRID_PASSTHROUGH = z'20'

5.1.1 Class Documentation

5.1.1.1 module asagi

ASAGI Fortran Interface.

Public Member Functions

- integer(kind=c_int) function grid_create_c (grid_type, hint, levels)
- integer(kind=c_int) function grid_set_comm (grid_id, comm)
- real(kind=c_double) function grid_min_x (grid_id)
- real(kind=c_double) function grid_max_x (grid_id)
- real(kind=c_double) function grid_min_y (grid_id)
- real(kind=c_double) function grid_max_y (grid_id)
- real(kind=c_double) function grid_min_z (grid_id)
- real(kind=c_double) function grid_max_z (grid_id)
- real(kind=c_double) function grid_delta_x (grid_id)
- real(kind=c_double) function grid_delta_y (grid_id)
- real(kind=c_double) function grid_delta_z (grid_id)
- integer(kind=c_int) function grid_var_size (grid_id)
- subroutine grid_close (grid_id)
- integer function grid_create (grid_type, hint, levels)
- integer function grid_create_array (grid_basictype, hint, levels)
- integer function grid_create_struct (count, block_length, displacments, types, hint, levels)
- integer function grid_set_param (grid_id, name, value, level)
- integer function grid_open (grid_id, filename, level)
- character function grid_get_byte_1d (grid_id, x, level)
- integer function grid_get_int_1d (grid_id, x, level)
- integer(kind=c_long) function grid_get_long_1d (grid_id, x, level)
- real function grid_get_float_1d (grid_id, x, level)
- real(kind=c_double) function grid_get_double_1d (grid_id, x, level)
- subroutine grid_get_buf_1d (grid_id, buf, x, level)
- character function grid_get_byte_2d (grid_id, x, y, level)
- integer function grid_get_int_2d (grid_id, x, y, level)
- integer(kind=c_long) function grid_get_long_2d (grid_id, x, y, level)
- real function grid_get_float_2d (grid_id, x, y, level)
- real(kind=c_double) function grid_get_double_2d (grid_id, x, y, level)
- subroutine grid_get_buf_2d (grid_id, buf, x, y, level)
- character function grid_get_byte_3d (grid_id, x, y, z, level)
- integer function grid_get_int_3d (grid_id, x, y, z, level)
- integer(kind=c_long) function grid_get_long_3d (grid_id, x, y, z, level)
- real function grid_get_float_3d (grid_id, x, y, z, level)
- real(kind=c_double) function grid_get_double_3d (grid_id, x, y, z, level)
- subroutine grid_get_buf_3d (grid_id, buf, x, y, z, level)

Public Attributes

- integer, parameter GRID_NO_HINT = z'0'
- integer, parameter GRID_HAS_TIME = z'1'
- integer, parameter GRID_NOMPI = z'2'
- integer, parameter SMALL_CACHE = z'4'
- integer, parameter GRID_LARGE_GRID = z'8'
- integer, parameter GRID_ADAPTIVE = z'10'
- integer, parameter GRID_PASSTHROUGH = z'20'

5.1.1.2 interface asagi::grid_get_byte

Interface for arbitrary dimensions.

Public Member Functions

- character function grid_get_byte_1d (grid_id, x, level)
- character function grid_get_byte_2d (grid_id, x, y, level)
- character function grid_get_byte_3d (grid_id, x, y, z, level)

Member Function Documentation

5.1.1.2.1 character function **asagi::grid_get_byte::grid_get_byte_1d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* integer, intent(in), optional *level*)

See also

asagi::Grid::getByte1D()

5.1.1.2.2 character function **asagi::grid_get_byte::grid_get_byte_2d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* integer, intent(in), optional *level*)

See also

asagi::Grid::getByte2D()

5.1.1.2.3 character function **asagi::grid_get_byte::grid_get_byte_3d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* real(kind=c_double), intent(in) *z,* integer, intent(in), optional *level*)

See also

asagi::Grid::getByte3D()

5.1.1.3 interface asagi::grid_get_int

Interface for arbitrary dimensions.

Public Member Functions

- integer function grid_get_int_1d (grid_id, x, level)
- integer function grid_get_int_2d (grid_id, x, y, level)
- integer function grid_get_int_3d (grid_id, x, y, z, level)

Member Function Documentation

5.1.1.3.1 integer function **asagi::grid_get_int::grid_get_int_1d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* integer, intent(in), optional *level*)

See also

asagi::Grid::getInt1D()

5.1.1.3.2 integer function **asagi::grid_get_int::grid_get_int_2d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* integer, intent(in), optional *level*)

See also

asagi::Grid::getInt2D()

5.1.1.3.3 integer function **asagi::grid_get_int::grid_get_int_3d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* real(kind=c_double), intent(in) *z,* integer, intent(in), optional *level*)

See also

asagi::Grid::getInt3D()

5.1.1.4 interface asagi::grid_get_long

Interface for arbitrary dimensions.

Public Member Functions

- integer(kind=c_long) function grid_get_long_1d (grid_id, x, level)
- integer(kind=c_long) function grid_get_long_2d (grid_id, x, y, level)
- integer(kind=c_long) function grid_get_long_3d (grid_id, x, y, z, level)

Member Function Documentation

5.1.1.4.1 integer(kind=c_long) function **asagi::grid_get_long::grid_get_long_1d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* integer, intent(in), optional *level*)

See also

asagi::Grid::getLong1D()

5.1.1.4.2 integer(kind=c_long) function **asagi::grid_get_long::grid_get_long_2d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* integer, intent(in), optional *level*)

See also

asagi::Grid::getLong2D()

5.1.1.4.3 integer(kind=c_long) function **asagi::grid_get_long::grid_get_long_3d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* real(kind=c_double), intent(in) *z,* integer, intent(in), optional *level*)

See also

asagi::Grid::getLong3D()

5.1.1.5 interface asagi::grid_get_float

Interface for arbitrary dimensions.

Public Member Functions

- real function grid_get_float_1d (grid_id, x, level)
- real function grid_get_float_2d (grid_id, x, y, level)
- real function grid_get_float_3d (grid_id, x, y, z, level)

Member Function Documentation

5.1.1.5.1 real function **asagi::grid_get_float::grid_get_float_1d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* integer, intent(in), optional *level*)

See also

asagi::Grid::getFloat1D()

5.1.1.5.2 real function **asagi::grid_get_float::grid_get_float_2d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* integer, intent(in), optional *level*)

See also

asagi::Grid::getFloat2D()

5.1.1.5.3 real function **asagi::grid_get_float::grid_get_float_3d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* real(kind=c_double), intent(in) *z,* integer, intent(in), optional *level*)

See also

asagi::Grid::getFloat3D()

5.1.1.6 interface asagi::grid_get_double

Interface for arbitrary dimensions.

Public Member Functions

- real(kind=c_double) function grid_get_double_1d (grid_id, x, level)
- real(kind=c_double) function grid_get_double_2d (grid_id, x, y, level)
- real(kind=c_double) function grid_get_double_3d (grid_id, x, y, z, level)

Member Function Documentation

5.1.1.6.1 real(kind=c_double) function asagi::grid_get_double::grid_get_double_1d (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* integer, intent(in), optional *level*)

See also

asagi::Grid::getDouble1D()

5.1.1.6.2 real(kind=c_double) function asagi::grid_get_double::grid_get_double_2d (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* integer, intent(in), optional *level*)

See also

asagi::Grid::getDouble2D()

5.1.1.6.3 real(kind=c_double) function asagi::grid_get_double::grid_get_double_3d (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* real(kind=c_double), intent(in) *z,* integer, intent(in), optional *level*)

See also

asagi::Grid::getDouble3D()

5.1.1.7 interface asagi::grid_get_buf

Interface for arbitrary dimensions.

Public Member Functions

- subroutine grid_get_buf_1d (grid_id, buf, x, level)
- subroutine grid_get_buf_2d (grid_id, buf, x, y, level)
- subroutine grid_get_buf_3d (grid_id, buf, x, y, z, level)

Member Function Documentation

5.1.1.7.1 subroutine asagi::grid_get_buf::grid_get_buf_1d (integer, intent(in) *grid_id,* type(c_ptr) *buf,* real(kind=c_double), intent(in) *x,* integer, intent(in), optional *level*)

See also

asagi::Grid::getBuf1D()

5.1.1.7.2 subroutine **asagi::grid_get_buf::grid_get_buf_2d** (integer, intent(in) *grid_id*, type(c_ptr) *buf*, real(kind=c_double), intent(in) *x*, real(kind=c_double), intent(in) *y*, integer, intent(in), optional *level*)

See also

asagi::Grid::getBuf2D()

5.1.1.7.3 subroutine **asagi::grid_get_buf::grid_get_buf_3d** (integer, intent(in) *grid_id*, type(c_ptr) *buf*, real(kind=c_double), intent(in) *x*, real(kind=c_double), intent(in) *y*, real(kind=c_double), intent(in) *z*, integer, intent(in), optional *level*)

See also

asagi::Grid::getBuf3D()

5.1.2 Function Documentation

5.1.2.1 subroutine **asagi::grid_close** (integer(kind=c_int) *grid_id*)

See also

asagi::Grid::close(asagi::Grid∗)

5.1.2.2 integer function **asagi::grid_create** (integer, intent(in), optional *grid_type*, integer, intent(in), optional *hint*, integer, intent(in), optional *levels*)

See also

asagi::Grid::create()

5.1.2.3 integer function **asagi::grid_create_array** (integer, intent(in), optional *grid_basictype*, integer, intent(in), optional *hint*, integer, intent(in), optional *levels*)

See also

asagi::Grid::createArray()

5.1.2.4 integer(kind=c_int) function **asagi::grid_create_c** (integer(kind=c_int) *grid_type*, integer(kind=c_int) *hint*, integer(kind=c_int) *levels*)

See also

asagi::Grid::Type
asagi::Grid::Error

5.1.2.5 integer function **asagi::grid_create_struct** (integer, intent(in) *count,* integer, dimension(∗), intent(in) *block_length,* integer(kind=c_long), dimension(∗), intent(in) *displacments,* integer, dimension(∗), intent(in) *types,* integer, intent(in), optional *hint,* integer, intent(in), optional *levels*)

See also

asagi::Grid::createStruct()

5.1.2.6 real(kind=c_double) function **asagi::grid_delta_x** (integer(kind=c_int) *grid_id*)

See also

asagi::Grid::getXDelta()

5.1.2.7 real(kind=c_double) function **asagi::grid_delta_y** (integer(kind=c_int) *grid_id*)

See also

asagi::Grid::getYDelta()

5.1.2.8 real(kind=c_double) function **asagi::grid_delta_z** (integer(kind=c_int) *grid_id*)

See also

asagi::Grid::getZDelta()

5.1.2.9 subroutine **asagi::grid_get_buf_1d** (integer, intent(in) *grid_id,* type(c_ptr) *buf,* real(kind=c_double), intent(in) *x,* integer, intent(in), optional *level*)

See also

asagi::Grid::getBuf1D()

5.1.2.10 subroutine **asagi::grid_get_buf_2d** (integer, intent(in) *grid_id,* type(c_ptr) *buf,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* integer, intent(in), optional *level*)

See also

asagi::Grid::getBuf2D()

5.1.2.11 subroutine **asagi::grid_get_buf_3d** (integer, intent(in) *grid_id,* type(c_ptr) *buf,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* real(kind=c_double), intent(in) *z,* integer, intent(in), optional *level*)

See also

asagi::Grid::getBuf3D()

5.1.2.12 character function **asagi::grid_get_byte_1d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* integer, intent(in), optional *level*)

See also

asagi::Grid::getByte1D()

5.1.2.13 character function **asagi::grid_get_byte_2d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* integer, intent(in), optional *level*)

See also

asagi::Grid::getByte2D()

5.1.2.14 character function **asagi::grid_get_byte_3d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* real(kind=c_double), intent(in) *z,* integer, intent(in), optional *level*)

See also

asagi::Grid::getByte3D()

5.1.2.15 real(kind=c_double) function **asagi::grid_get_double_1d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* integer, intent(in), optional *level*)

See also

asagi::Grid::getDouble1D()

5.1.2.16 real(kind=c_double) function **asagi::grid_get_double_2d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* integer, intent(in), optional *level*)

See also

asagi::Grid::getDouble2D()

5.1.2.17 real(kind=c_double) function **asagi::grid_get_double_3d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* real(kind=c_double), intent(in) *z,* integer, intent(in), optional *level*)

See also

asagi::Grid::getDouble3D()

5.1.2.18 real function **asagi::grid_get_float_1d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* integer, intent(in), optional *level*)

See also

asagi::Grid::getFloat1D()

5.1.2.19 real function **asagi::grid_get_float_2d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* integer, intent(in), optional *level*)

See also

asagi::Grid::getFloat2D()

5.1.2.20 real function **asagi::grid_get_float_3d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* real(kind=c_double), intent(in) *z,* integer, intent(in), optional *level*)

See also

asagi::Grid::getFloat3D()

5.1.2.21 integer function **asagi::grid_get_int_1d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* integer, intent(in), optional *level*)

See also

asagi::Grid::getInt1D()

5.1.2.22 integer function **asagi::grid_get_int_2d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* integer, intent(in), optional *level*)

See also

asagi::Grid::getInt2D()

5.1.2.23 integer function **asagi::grid_get_int_3d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* real(kind=c_double), intent(in) *z,* integer, intent(in), optional *level*)

See also

asagi::Grid::getInt3D()

5.1.2.24 integer(kind=c_long) function **asagi::grid_get_long_1d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* integer, intent(in), optional *level*)

See also

asagi::Grid::getLong1D()

5.1.2.25 integer(kind=c_long) function **asagi::grid_get_long_2d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* integer, intent(in), optional *level*)

See also

asagi::Grid::getLong2D()

5.1.2.26 integer(kind=c_long) function **asagi::grid_get_long_3d** (integer, intent(in) *grid_id,* real(kind=c_double), intent(in) *x,* real(kind=c_double), intent(in) *y,* real(kind=c_double), intent(in) *z,* integer, intent(in), optional *level*)

See also

asagi::Grid::getLong3D()

5.1.2.27 real(kind=c_double) function **asagi::grid_max_x** (integer(kind=c_int) *grid_id*)

See also

asagi::Grid::getXMax()

5.1.2.28 real(kind=c_double) function **asagi::grid_max_y** (integer(kind=c_int) *grid_id*)

See also

asagi::Grid::getYMax()

5.1.2.29 real(kind=c_double) function **asagi::grid_max_z** (integer(kind=c_int) *grid_id*)

See also

asagi::Grid::getZMax()

5.1.2.30 real(kind=c_double) function **asagi::grid_min_x** (integer(kind=c_int) *grid_id*)

See also

asagi::Grid::getXMin()

5.1.2.31 real(kind=c_double) function **asagi::grid_min_y** (integer(kind=c_int) *grid_id*)

See also

asagi::Grid::getYMin()

5.1.2.32 real(kind=c_double) function **asagi::grid_min_z** (integer(kind=c_int) *grid_id*)

See also

asagi::Grid::getZMin()

5.1.2.33 integer function **asagi::grid_open** (integer, intent(in) *grid_id,* character∗(∗), intent(in) *filename,* integer, intent(in), optional *level*)

See also

asagi::Grid::open()

5.1.2.34 integer(kind=c_int) function **asagi::grid_set_comm** (integer(kind=c_int) *grid_id*, integer(kind=c_int) *comm*)

See also

asagi::Grid::setComm()

5.1.2.35 integer function **asagi::grid_set_param** (integer, intent(in) *grid_id*, character∗(∗), intent(in) *name*, character∗(∗), intent(in) *value*, integer, intent(in), optional *level*)

See also

asagi::Grid::setParam()

5.1.2.36 integer(kind=c_int) function **asagi::grid_var_size** (integer(kind=c_int) *grid_id*)

See also

asagi::Grid::getVarSize()

5.1.3 Variable Documentation

5.1.3.1 integer, parameter **asagi::GRID_ADAPTIVE** = z'10'

See also

asagi::Grid::ADAPTIVE

5.1.3.2 integer, parameter **asagi::GRID_HAS_TIME** = z'1'

See also

asagi::Grid::HAS_TIME

5.1.3.3 integer, parameter **asagi::GRID_LARGE_GRID** = z'8'

See also

asagi::Grid::LARGE_GRID

5.1.3.4 integer, parameter **asagi::GRID_NO_HINT** = z'0'

See also

asagi::Grid::NO_HINT

5.1.3.5 integer, parameter asagi::GRID_NOMPI = z'2'

See also

asagi::Grid::NOMPI

5.1.3.6 integer, parameter asagi::GRID_PASSTHROUGH = z'20'

See also

asagi::Grid::PASS_THROUGH

5.1.3.7 integer, parameter asagi::SMALL_CACHE = z'4'

See also

asagi::Grid::SMALL_CACHE

5.2 C Interface

Typedefs

- typedef asagi::Grid grid_handle

Enumerations

- enum grid_type
- enum grid_error

Functions

- grid_handle ∗ grid_create (grid_type type, unsigned int hint, unsigned int levels)
- grid_handle ∗ grid_create_array (grid_type basic_type, unsigned int hint, unsigned int levels)
- grid_handle ∗ grid_create_struct (unsigned int count, unsigned int blockLength[ ], unsigned long displacements[ ], grid_type types[ ], unsigned int hint, unsigned int levels)
- grid_error grid_set_comm (grid_handle ∗handle, MPI_Comm comm)
- grid_error grid_set_param (grid_handle ∗handle, const char ∗name, const char ∗value, unsigned int level)
- grid_error grid_open (grid_handle ∗handle, const char ∗filename, unsigned int level)
- double grid_min_x (grid_handle ∗handle)
- double grid_min_y (grid_handle ∗handle)
- double grid_min_z (grid_handle ∗handle)
- double grid_max_x (grid_handle ∗handle)
- double grid_max_y (grid_handle ∗handle)
- double grid_max_z (grid_handle ∗handle)
- double grid_delta_x (grid_handle ∗handle)
- double grid_delta_y (grid_handle ∗handle)
- double grid_delta_z (grid_handle ∗handle)
- unsigned int grid_var_size (grid_handle ∗handle)
- unsigned char grid_get_byte_1d (grid_handle ∗handle, double x, unsigned int level)
- int grid_get_int_1d (grid_handle ∗handle, double x, unsigned int level)
- long grid_get_long_1d (grid_handle ∗handle, double x, unsigned int level)
- float grid_get_float_1d (grid_handle ∗handle, double x, unsigned int level)
- double grid_get_double_1d (grid_handle ∗handle, double x, unsigned int level)
- void grid_get_buf_1d (grid_handle ∗handle, void ∗buf, double x, unsigned int level)
- unsigned char grid_get_byte_2d (grid_handle ∗handle, double x, double y, unsigned int level)
- int grid_get_int_2d (grid_handle ∗handle, double x, double y, unsigned int level)
- long grid_get_long_2d (grid_handle ∗handle, double x, double y, unsigned int level)

- float grid_get_float_2d (grid_handle ∗handle, double x, double y, unsigned int level)
- double grid_get_double_2d (grid_handle ∗handle, double x, double y, unsigned int level)
- void grid_get_buf_2d (grid_handle ∗handle, void ∗buf, double x, double y, unsigned int level)
- unsigned char grid_get_byte_3d (grid_handle ∗handle, double x, double y, double z, unsigned int level)
- int grid_get_int_3d (grid_handle ∗handle, double x, double y, double z, unsigned int level)
- long grid_get_long_3d (grid_handle ∗handle, double x, double y, double z, unsigned int level)
- float grid_get_float_3d (grid_handle ∗handle, double x, double y, double z, unsigned int level)
- double grid_get_double_3d (grid_handle ∗handle, double x, double y, double z, unsigned int level)
- void grid_get_buf_3d (grid_handle ∗handle, void ∗buf, double x, double y, double z, unsigned int level)
- void grid_close (grid_handle ∗handle)

Variables

- const unsigned int GRID_NO_HINT = 0x0
- const unsigned int GRID_HAS_TIME = 0x1
- const unsigned int GRID_NOMPI = 0x2
- const unsigned int SMALL_CACHE = 0x4
- const unsigned int GRID_LARGE_GRID = 0x8
- const unsigned int GRID_ADAPTIVE = 0x10
- const unsigned int PASS_THROUGH = 0x20

5.2.1 Typedef Documentation

5.2.1.1 typedef struct **grid_handle grid_handle**

A handle for a grid

5.2.2 Enumeration Type Documentation

5.2.2.1 enum **grid_error**

See also

asagi::Grid::Error

5.2.2.2 enum **grid_type**

See also

asagi::Grid::Type

5.2.3 Function Documentation

5.2.3.1 void **grid_close** (**grid_handle** ∗ *handle*)

See also

asagi::Grid::close(asagi::Grid∗)

5.2.3.2 **grid_handle**∗ **grid_create** (**grid_type** *type,* unsigned int *hint,* unsigned int *levels*)

See also

asagi::Grid::create()

5.2.3.3 **grid_handle**∗ **grid_create_array** (**grid_type** *basic_type,* unsigned int *hint,* unsigned int *levels*)

See also

asagi::Grid::createArray()

5.2.3.4 **grid_handle**∗ **grid_create_struct** (unsigned int *count,* unsigned int *blockLength[ ],* unsigned long *displacements[ ],* **grid_type** *types[ ],* unsigned int *hint,* unsigned int *levels*)

See also

asagi::Grid::createStruct()

5.2.3.5 double **grid_delta_x** (**grid_handle** ∗ *handle*)

See also

asagi::Grid::getXDelta()

5.2.3.6 double **grid_delta_y** (**grid_handle** ∗ *handle*)

See also

asagi::Grid::getYDelta()

5.2.3.7 double **grid_delta_z** (**grid_handle** ∗ *handle*)

See also

asagi::Grid::getZDelta()

5.2.3.8 void **grid_get_buf_1d** (**grid_handle** ∗ *handle,* void ∗ *buf,* double *x,* unsigned int *level*)

See also

asagi::Grid::getBuf1D()

5.2.3.9 void **grid_get_buf_2d** (**grid_handle** ∗ *handle,* void ∗ *buf,* double *x,* double *y,* unsigned int *level*)

See also

asagi::Grid::getBuf2D()

5.2.3.10 void **grid_get_buf_3d** (**grid_handle** ∗ *handle,* void ∗ *buf,* double *x,* double *y,* double *z,* unsigned int *level*)

See also

asagi::Grid::getBuf3D()

5.2.3.11 unsigned char **grid_get_byte_1d** (**grid_handle** ∗ *handle,* double *x,* unsigned int *level*)

See also

asagi::Grid::getByte1D()

5.2.3.12 unsigned char **grid_get_byte_2d** (**grid_handle** ∗ *handle,* double *x,* double *y,* unsigned int *level*)

See also

asagi::Grid::getByte2D()

5.2.3.13 unsigned char **grid_get_byte_3d** (**grid_handle** ∗ *handle,* double *x,* double *y,* double *z,* unsigned int *level*)

See also

asagi::Grid::getByte3D()

5.2.3.14 double **grid_get_double_1d** (**grid_handle** ∗ *handle,* double *x,* unsigned int *level*)

See also

asagi::Grid::getDouble1D()

5.2.3.15 double **grid_get_double_2d** (**grid_handle** $*$ *handle*, double *x*, double *y*, unsigned int *level*)

See also

asagi::Grid::getDouble2D()

5.2.3.16 double **grid_get_double_3d** (**grid_handle** $*$ *handle*, double *x*, double *y*, double *z*, unsigned int *level*)

See also

asagi::Grid::getDouble3D()

5.2.3.17 float **grid_get_float_1d** (**grid_handle** $*$ *handle*, double *x*, unsigned int *level*)

See also

asagi::Grid::getFloat1D()

5.2.3.18 float **grid_get_float_2d** (**grid_handle** $*$ *handle*, double *x*, double *y*, unsigned int *level*)

See also

asagi::Grid::getFloat2D()

5.2.3.19 float **grid_get_float_3d** (**grid_handle** $*$ *handle*, double *x*, double *y*, double *z*, unsigned int *level*)

See also

asagi::Grid::getFloat3D()

5.2.3.20 int **grid_get_int_1d** (**grid_handle** $*$ *handle*, double *x*, unsigned int *level*)

See also

asagi::Grid::getInt1D()

5.2.3.21 int **grid_get_int_2d** (**grid_handle** $*$ *handle*, double *x*, double *y*, unsigned int *level*)

See also

asagi::Grid::getInt2D()

5.2.3.22 int **grid_get_int_3d** (**grid_handle** $*$ *handle*, double *x*, double *y*, double *z*, unsigned int *level*)

See also

asagi::Grid::getInt3D()

5.2.3.23 long **grid_get_long_1d** (**grid_handle** ∗ *handle*, double *x*, unsigned int *level*)

See also

asagi::Grid::getLong1D()

5.2.3.24 long **grid_get_long_2d** (**grid_handle** ∗ *handle*, double *x*, double *y*, unsigned int *level*)

See also

asagi::Grid::getLong2D()

5.2.3.25 long **grid_get_long_3d** (**grid_handle** ∗ *handle*, double *x*, double *y*, double *z*, unsigned int *level*)

See also

asagi::Grid::getLong3D()

5.2.3.26 double **grid_max_x** (**grid_handle** ∗ *handle*)

See also

asagi::Grid::getYMax()

5.2.3.27 double **grid_max_y** (**grid_handle** ∗ *handle*)

See also

asagi::Grid::getZMin()

5.2.3.28 double **grid_max_z** (**grid_handle** ∗ *handle*)

See also

asagi::Grid::getZMax()

5.2.3.29 double **grid_min_x** (**grid_handle** ∗ *handle*)

See also

asagi::Grid::getXMin()

5.2.3.30 double **grid_min_y** (**grid_handle** ∗ *handle*)

See also

asagi::Grid::getXMax()

5.2.3.31 double **grid_min_z** (**grid_handle** ∗ *handle*)

See also

asagi::Grid::getYMin()

5.2.3.32 **grid_error grid_open** (**grid_handle** ∗ *handle,* const char ∗ *filename,* unsigned int *level*)

See also

asagi::Grid::open()

5.2.3.33 **grid_error grid_set_comm** (**grid_handle** ∗ *handle,* MPI_Comm *comm*)

See also

asagi::Grid::setComm()

5.2.3.34 **grid_error grid_set_param** (**grid_handle** ∗ *handle,* const char ∗ *name,* const char ∗ *value,* unsigned int *level*)

See also

asagi::Grid::setParam()

5.2.3.35 unsigned int **grid_var_size** (**grid_handle** ∗ *handle*)

See also

asagi::Grid::getVarSize()

5.2.4 Variable Documentation

5.2.4.1 const unsigned int **GRID_ADAPTIVE** = 0x10

See also

asagi::Grid::ADAPTIVE

5.2.4.2 const unsigned int **GRID_HAS_TIME** = 0x1

See also

asagi::Grid::HAS_TIME

5.2.4.3 const unsigned int **GRID_LARGE_GRID** = 0x8

See also

asagi::Grid::LARGE_GRID

5.2.4.4 const unsigned int **GRID_NO_HINT** = 0x0

See also

asagi::Grid::NO_HINT

5.2.4.5 const unsigned int **GRID_NOMPI** = 0x2

See also

asagi::Grid::NOMPI

5.2.4.6 const unsigned int **PASS_THROUGH** = 0x20

See also

asagi::Grid::PASS_THROUGH

5.2.4.7 const unsigned int **SMALL_CACHE** = 0x4

See also

asagi::Grid::SMALL_CACHE

5.3 C++ Interface

Classes

- class asagi::Grid

  C++ Interface for ASAGI. More...

Enumerations

- enum asagi::Grid::Type { asagi::Grid::BYTE, asagi::Grid::INT, asagi::Grid::LON-G, asagi::Grid::FLOAT, asagi::Grid::DOUBLE }
- enum asagi::Grid::Error { asagi::Grid::SUCCESS = 0, asagi::Grid::MPI_ERRO-R, asagi::Grid::UNKNOWN_PARAM, asagi::Grid::INVALID_VALUE, asagi::Grid-::NOT_OPEN, asagi::Grid::VAR_NOT_FOUND, asagi::Grid::UNSUPPORTED_-DIMENSIONS, asagi::Grid::MULTIPLE_TOPGRIDS, asagi::Grid::INVALID_VA-R_SIZE }

Functions

- virtual asagi::Grid::∼Grid ()
- virtual Error asagi::Grid::setComm (MPI_Comm comm)=0
- virtual Error asagi::Grid::setParam (const char ∗name, const char ∗value, unsigned int level=0)=0
- virtual Error asagi::Grid::open (const char ∗filename, unsigned int level=0)=0
- virtual double asagi::Grid::getXMin () const =0
- virtual double asagi::Grid::getYMin () const =0
- virtual double asagi::Grid::getZMin () const =0
- virtual double asagi::Grid::getXMax () const =0
- virtual double asagi::Grid::getYMax () const =0
- virtual double asagi::Grid::getZMax () const =0
- virtual double asagi::Grid::getXDelta () const =0
- virtual double asagi::Grid::getYDelta () const =0
- virtual double asagi::Grid::getZDelta () const =0
- virtual unsigned int asagi::Grid::getVarSize () const =0
- virtual unsigned char asagi::Grid::getByte1D (double x, unsigned int level=0)=0
- virtual int asagi::Grid::getInt1D (double x, unsigned int level=0)=0
- virtual long asagi::Grid::getLong1D (double x, unsigned int level=0)=0
- virtual float asagi::Grid::getFloat1D (double x, unsigned int level=0)=0
- virtual double asagi::Grid::getDouble1D (double x, unsigned int level=0)=0
- virtual void asagi::Grid::getBuf1D (void ∗buf, double x, unsigned int level=0)=0
- virtual unsigned char asagi::Grid::getByte2D (double x, double y, unsigned int level=0)=0
- virtual int asagi::Grid::getInt2D (double x, double y, unsigned int level=0)=0
- virtual long asagi::Grid::getLong2D (double x, double y, unsigned int level=0)=0
- virtual float asagi::Grid::getFloat2D (double x, double y, unsigned int level=0)=0

- virtual double asagi::Grid::getDouble2D (double x, double y, unsigned int level=0)=0
- virtual void asagi::Grid::getBuf2D (void ∗buf, double x, double y, unsigned int level=0)=0
- virtual unsigned char asagi::Grid::getByte3D (double x, double y, double z, unsigned int level=0)=0
- virtual int asagi::Grid::getInt3D (double x, double y, double z, unsigned int level=0)=0
- virtual long asagi::Grid::getLong3D (double x, double y, double z, unsigned int level=0)=0
- virtual float asagi::Grid::getFloat3D (double x, double y, double z, unsigned int level=0)=0
- virtual double asagi::Grid::getDouble3D (double x, double y, double z, unsigned int level=0)=0
- virtual void asagi::Grid::getBuf3D (void ∗buf, double x, double y, double z, unsigned int level=0)=0
- virtual unsigned long asagi::Grid::getCounter (const char ∗name, unsigned int level=0)=0
- static asagi::Grid ∗ asagi::Grid::create (Type type=FLOAT, unsigned int hint=NO-_HINT, unsigned int levels=1)
- static asagi::Grid ∗ asagi::Grid::createArray (Type basicType=FLOAT, unsigned int hint=NO_HINT, unsigned int levels=1)
- static asagi::Grid ∗ asagi::Grid::createStruct (unsigned int count, unsigned int blockLength[], unsigned long displacements[], asagi::Grid::Type types[], unsigned int hint=NO_HINT, unsigned int levels=1)
- static void asagi::Grid::close (asagi::Grid ∗grid)

5.3.1 Class Documentation

5.3.1.1 class asagi::Grid

C++ Interface for ASAGI.

Public Types

- enum Type { BYTE, INT, LONG, FLOAT, DOUBLE }
- enum Error { SUCCESS = 0, MPI_ERROR, UNKNOWN_PARAM, INVALID_V-ALUE, NOT_OPEN, VAR_NOT_FOUND, UNSUPPORTED_DIMENSIONS, M-ULTIPLE_TOPGRIDS, INVALID_VAR_SIZE }

Public Member Functions

- virtual ∼Grid ()
- virtual Error setComm (MPI_Comm comm)=0
- virtual Error setParam (const char ∗name, const char ∗value, unsigned int level=0)=0
- virtual Error open (const char ∗filename, unsigned int level=0)=0
- virtual double getXMin () const =0

- virtual double getYMin () const =0
- virtual double getZMin () const =0
- virtual double getXMax () const =0
- virtual double getYMax () const =0
- virtual double getZMax () const =0
- virtual double getXDelta () const =0
- virtual double getYDelta () const =0
- virtual double getZDelta () const =0
- virtual unsigned int getVarSize () const =0
- virtual unsigned char getByte1D (double x, unsigned int level=0)=0
- virtual int getInt1D (double x, unsigned int level=0)=0
- virtual long getLong1D (double x, unsigned int level=0)=0
- virtual float getFloat1D (double x, unsigned int level=0)=0
- virtual double getDouble1D (double x, unsigned int level=0)=0
- virtual void getBuf1D (void ∗buf, double x, unsigned int level=0)=0
- virtual unsigned char getByte2D (double x, double y, unsigned int level=0)=0
- virtual int getInt2D (double x, double y, unsigned int level=0)=0
- virtual long getLong2D (double x, double y, unsigned int level=0)=0
- virtual float getFloat2D (double x, double y, unsigned int level=0)=0
- virtual double getDouble2D (double x, double y, unsigned int level=0)=0
- virtual void getBuf2D (void ∗buf, double x, double y, unsigned int level=0)=0
- virtual unsigned char getByte3D (double x, double y, double z, unsigned int level=0)=0
- virtual int getInt3D (double x, double y, double z, unsigned int level=0)=0
- virtual long getLong3D (double x, double y, double z, unsigned int level=0)=0
- virtual float getFloat3D (double x, double y, double z, unsigned int level=0)=0
- virtual double getDouble3D (double x, double y, double z, unsigned int level=0)=0
- virtual void getBuf3D (void ∗buf, double x, double y, double z, unsigned int level=0)=0
- virtual bool exportPng (const char ∗filename, unsigned int level=0)=0
- virtual unsigned long getCounter (const char ∗name, unsigned int level=0)=0

Static Public Member Functions

- static asagi::Grid ∗ create (Type type=FLOAT, unsigned int hint=NO_HINT, unsigned int levels=1)
- static asagi::Grid ∗ createArray (Type basicType=FLOAT, unsigned int hint=NO_HINT, unsigned int levels=1)
- static asagi::Grid ∗ createStruct (unsigned int count, unsigned int blockLength[], unsigned long displacements[], asagi::Grid::Type types[], unsigned int hint=NO_HINT, unsigned int levels=1)
- static void close (asagi::Grid ∗grid)

Static Public Attributes

- static const unsigned int NO_HINT = 0x0
- static const unsigned int HAS_TIME = 0x1
- static const unsigned int NOMPI = 0x2
- static const unsigned int SMALL_CACHE = 0x4
- static const unsigned int LARGE_GRID = 0x8
- static const unsigned int ADAPTIVE = 0x10
- static const unsigned int PASS_THROUGH = 0x20

Member Function Documentation

5.3.1.1.1 virtual bool **asagi::Grid::exportPng** (const char * *filename,* unsigned int *level =* 0) `[pure virtual]`

Exports the grid to png file. Should not be used for 3D grids

Member Data Documentation

5.3.1.1.2 const unsigned int **asagi::Grid::ADAPTIVE** = 0x10 `[static]`

Use an adaptive container. Allows you to load multiple grids with the same level of detail. Not fully tested yet.

5.3.1.1.3 const unsigned int **asagi::Grid::HAS_TIME** = 0x1 `[static]`

One dimension in the grid is a time dimension

5.3.1.1.4 const unsigned int **asagi::Grid::LARGE_GRID** = 0x8 `[static]`

Use this, if you are going to load a large grid with ASAGI

5.3.1.1.5 const unsigned int **asagi::Grid::NO_HINT** = 0x0 `[static]`

Does not provide any hint for ASAGI (default)

5.3.1.1.6 const unsigned int **asagi::Grid::NOMPI** = 0x2 `[static]`

Don’t use any MPI, even when compiled with MPI support (MPI_Init may not be called before creating the grid)

5.3.1.1.7 const unsigned int **asagi::Grid::PASS_THROUGH** = 0x20 `[static]`

Use ASAGI only as a wrapper for the underlying I/O library. ASAGI does not cache any values. Works also without MPI.

5.3.1.1.8 const unsigned int **asagi::Grid::SMALL_CACHE** = 0x4 `[static]`

ASAGI should use a small cache. Less memory is used, but more cache misses occur.

5.3.2 Enumeration Type Documentation

5.3.2.1 enum **asagi::Grid::Error**

Possible errors that could occure

Enumerator:

SUCCESS	No error
MPI_ERROR	An MPI function failed
UNKNOWN_PARAM	Unknown configuration parameter
INVALID_VALUE	Invalid configuration value
NOT_OPEN	Could not open input file
VAR_NOT_FOUND	netCDF variable not found
UNSUPPORTED_DIMENSIONS	Unsupported number of dimensions input file
MULTIPLE_TOPGRIDS	More than one topmost grid specified
INVALID_VAR_SIZE	Variable size in the input file does not match the type

5.3.2.2 enum **asagi::Grid::Type**

The basic types supported by ASAGI

Enumerator:

BYTE	signed byte
INT	signed 4-byte integer
LONG	signed 8-byte integer
FLOAT	4-byte floating point value
DOUBLE	8-byte floating point value

5.3.3 Function Documentation

5.3.3.1 static void **asagi::Grid::close (asagi::Grid ∗ *grid*)** `[inline, static]`

Frees all memory resources assciated with `grid`. After a grid is closed you cannot access any values and you can not reopen another NetCDF file.

This function does the same as calling `delete grid;` and it is the C++ equivalent to grid_close(grid_handle∗) and ASAGI::grid_close

Parameters

grid	The grid that should be closed.

5.3.3.2 static **asagi::Grid∗ asagi::Grid::create (Type** *type* **= FLOAT,** unsigned int *hint* **= NO_HINT,** unsigned int *levels* **=** `1` **)** `[static]`

Creates a new grid with basic values

Parameters

type	The type of the grid
hint	A combination of hints
levels	The number of level this grid should have

5.3.3.3 static asagi::Grid∗ asagi::Grid::createArray (Type *basicType* = FLOAT, unsigned int *hint* = NO_HINT, unsigned int *levels* = 1) [static]

Creates a new grid with array values

Parameters

basicType	The type of the array values in the grid
hint	A combination of hints
levels	The number of levels this grid should have

5.3.3.4 static asagi::Grid∗ asagi::Grid::createStruct (unsigned int *count*, unsigned int *blockLength[ ]*, unsigned long *displacements[ ]*, asagi::Grid::Type *types[ ]*, unsigned int *hint* = NO_HINT, unsigned int *levels* = 1) [static]

Creates a new grid with a struct, very similar to MPI_Type_create_struct

Parameters

count	Number of blocks in the struct
blockLength	Number of elements in each block
displace-ments	Displacement of each block
types	Block type
hint	A combination of hints
levels	The number of levels this grid should have

5.3.3.5 virtual void asagi::Grid::getBuf1D (void ∗ *buf*, double *x*, unsigned int *level* = 0) [pure virtual]

See also

getBuf2D

5.3.3.6 virtual void asagi::Grid::getBuf2D (void ∗ *buf*, double *x*, double *y*, unsigned int *level* = 0) [pure virtual]

Copys the element at (x,y) into buf. The buffer size has to be (at least) getVarSize() bytes.

5.3.3.7 virtual void asagi::Grid::getBuf3D (void ∗ *buf*, double *x*, double *y*, double *z*, unsigned int *level* = 0) [pure virtual]

See also

getBuf2D

5.3.3.8 virtual unsigned char asagi::Grid::getByte1D (double *x*, unsigned int *level* = 0) `[pure virtual]`

See also

getByte2D

5.3.3.9 virtual unsigned char asagi::Grid::getByte2D (double *x*, double *y*, unsigned int *level* = 0) `[pure virtual]`

If the grid contains array values, only the first element of the array is returned

Returns

The element at (x,y) as a char

5.3.3.10 virtual unsigned char asagi::Grid::getByte3D (double *x*, double *y*, double *z*, unsigned int *level* = 0) `[pure virtual]`

See also

getByte2D

5.3.3.11 virtual unsigned long asagi::Grid::getCounter (const char ∗ *name*, unsigned int *level* = 0) `[pure virtual]`

Gets the current value of a counter for a grid level.

Possible counter names:

- **accesses** Total number of data accesses
- **mpi_transfers** Number of blocks transfered between processes
- **file_load** Number of blocks loaded from file (after initialization)
- **local_hits** Number values that where already in local memory
- **local_misses** Number of values that where not already in local memory

Returns

The current counter value or 0 if the name is not defined

Warning

The performance counters are not threadsafe for performance reason. You may get wrong result when using more than one thread.

5.3.3.12 virtual double asagi::Grid::getDouble1D (double *x*, unsigned int *level* = `0`) `[pure virtual]`

See also

getDouble2D

5.3.3.13 virtual double asagi::Grid::getDouble2D (double *x*, double *y*, unsigned int *level* = `0`) `[pure virtual]`

Returns

The element at (x,y) as a double

See also

getByte2D

5.3.3.14 virtual double asagi::Grid::getDouble3D (double *x*, double *y*, double *z*, unsigned int *level* = `0`) `[pure virtual]`

See also

getDouble2D

5.3.3.15 virtual float asagi::Grid::getFloat1D (double *x*, unsigned int *level* = `0`) `[pure virtual]`

See also

getFloat2D

5.3.3.16 virtual float asagi::Grid::getFloat2D (double *x*, double *y*, unsigned int *level* = `0`) `[pure virtual]`

Returns

The element at (x,y) as a float

See also

getByte2D

5.3.3.17 virtual float asagi::Grid::getFloat3D (double *x*, double *y*, double *z*, unsigned int *level* = `0`) `[pure virtual]`

See also

getFloat2D

5.3.3.18 virtual int **asagi::Grid::getInt1D** (double *x*, unsigned int *level* = `0`) `[pure virtual]`

See also

getInt2D

5.3.3.19 virtual int **asagi::Grid::getInt2D** (double *x*, double *y*, unsigned int *level* = `0`) `[pure virtual]`

Returns

The element at (x,y) as an integer

See also

getByte2D

5.3.3.20 virtual int **asagi::Grid::getInt3D** (double *x*, double *y*, double *z*, unsigned int *level* = `0`) `[pure virtual]`

See also

getInt2D

5.3.3.21 virtual long **asagi::Grid::getLong1D** (double *x*, unsigned int *level* = `0`) `[pure virtual]`

See also

getLong2D

5.3.3.22 virtual long **asagi::Grid::getLong2D** (double *x*, double *y*, unsigned int *level* = `0`) `[pure virtual]`

Returns

The element at (x,y) as a long

See also

getByte2D

5.3.3.23 virtual long **asagi::Grid::getLong3D** (double *x*, double *y*, double *z*, unsigned int *level* = `0`) `[pure virtual]`

See also

getLong2D

5.3.3.24 virtual unsigned int asagi::Grid::getVarSize ( ) const [pure virtual]

Returns

The number of bytes that are stored in each grid cell

5.3.3.25 virtual double asagi::Grid::getXDelta ( ) const [pure virtual]

Returns

The difference of two coordinates in x dimension

5.3.3.26 virtual double asagi::Grid::getXMax ( ) const [pure virtual]

Returns

The maxmium allowed coordinate in x dimension

5.3.3.27 virtual double asagi::Grid::getXMin ( ) const [pure virtual]

Returns

The minimum allowed coordinate in x dimension

5.3.3.28 virtual double asagi::Grid::getYDelta ( ) const [pure virtual]

Returns

The difference of two coordinates in y dimension

5.3.3.29 virtual double asagi::Grid::getYMax ( ) const [pure virtual]

Returns

The maximum allowed coordinate in y dimension

5.3.3.30 virtual double asagi::Grid::getYMin ( ) const [pure virtual]

Returns

The minimum allowed coordinate in y dimension

5.3.3.31 virtual double asagi::Grid::getZDelta ( ) const [pure virtual]

Returns

The difference of two coordinates in z dimension

5.3.3.32 virtual double asagi::Grid::getZMax ( ) const [pure virtual]

Returns

The maximum allowed coordinate in z dimension

5.3.3.33 virtual double **asagi::Grid::getZMin** ( ) const [pure virtual]

Returns

The minimum allowed coordinate in z dimension

5.3.3.34 virtual Error **asagi::Grid::open** (const char ∗ *filename*, unsigned int *level* = 0) [pure virtual]

Loads values from a NetCDF file.

This function must be called for each level of detail

5.3.3.35 virtual Error **asagi::Grid::setComm** (MPI_Comm *comm*) [pure virtual]

Call this function before open() if another communicator than the default MPI_COMM_-WORLD should be used.

5.3.3.36 virtual Error **asagi::Grid::setParam** (const char ∗ *name*, const char ∗ *value*, unsigned int *level* = 0) [pure virtual]

Changes a grid parameter.

This function allows you to change ASAGI's configuration. It must be called before calling open(const char∗, unsigned int).

The following parameters are supported:

- **value-position** The value should be either `cell-centered` (default) or `vertex-centered`.

  Note: This parameter does not depend on the level.
- **variable-name** The name of the variable in the NetCDF file (default: "z")
- **time-dimension** The dimension that holds the time. Only useful with the hint -HAS_TIME. Should be either "x", "y" or "z". (Default: the last dimension of the grid)
- **x-block-size** The block size in x dimension (Default: 50)
- **y-block-size** The block size in y dimension (Default: 50 or 1 if it is an 1-dimensional grid)
- **z-block-size** The block size in z dimension (Default: 50 or 1 if it is an 1- or 2-dimensional grid)
- **block-cache-size** Number of blocks cached on each node (Default: 80)
- **cache-hand-spread** The difference between the hands of the 2-handed clock algorithm (Default: block-cache-size/2)
- **multigrid-size** Sets the number of grids for the level. Call this before setting any other parameter. (Default 1)

Parameters

name	The name of the parameter
value	The new value for the parameter
level	Change the parameter for the specified level of detail. Should be 0 when setting **value-position**

Returns

`SUCCESS` if the parameter was successfully changed
`UNKNOWN_PARAM` if the parameter is not supported
`INVALID_VALUE` if the parameter does not accept this value

5.3.3.37 virtual asagi::Grid::∼Grid ( ) `[inline, virtual]`

See also

close(asagi::Grid∗)