# PIO API

This page has been moved, the new location is
http://code.google.com/p/parallelio/wiki/PIOAPI

## PIO specific interfaces

- subroutine PIO_init(comp_rank, comp_comm, num_iotasks, num_aggregator, stride, Rearranger, IOsystem, base)
  - integer(i4), intent(in) :: comp_rank
  - integer(i4), intent(in) :: comp_comm
  - integer(i4), intent(in) :: num_iotasks
  - integer(i4), intent(in) :: num_aggregator
  - integer(i4), intent(in) :: stride
  - integer, intent(in) :: Rearranger !defined in pio_types currently allowed values are:
    - PIO_rearr_none ! pio does no data rearrangment, data is assumed to be in it's final form when passed to pio
    - PIO_rearr_mct ! pio uses mct to rearrange the data from the computational layout to the io layout.
    - PIO_rearr_box ! pio uses an internal rearranger to rearrange the data from the computational layout to the io layout.
  - type (IOsystem_desc_t), intent(out) :: IOsystem ! IO descriptor to initalize
  - integer, optional :: base ! can be used to set the offset of the 0 rank of the io communicator within the comm communicator.
- subroutine PIO_initDecomp(Iosystem,baseTYPE,dims,lenBLOCKS,compDOF,ioDOFR,ioDOFW,start,cnt,IOdesc) PIO_initDecomp(Iosystem, baseTYPE,dims,lenBLOCKS,compDOF,ioDOFR,start,cnt,IOdesc) PIO_initDecomp(IOsystem,baseTYPE,dims,lenBLOCKS,compDOF,ioDOFR, ioDOFW,IOdesc) PIO_initDecomp(IOsystem,baseTYPE,dims,lenBLOCKS,compDOF,ioDOFR,IOdesc) PIO_initDecomp(IOSystem,baseType, dims,compDOF,IOdesc)
  - type (IOSystem_desc_t), intent(in) :: IOsystem
  - integer(i4), intent(in) :: baseTYPE ! type of array

    Unknown macro: {int,real4,real8}
  - integer(i4), intent(in) :: dims( ! global dimensions of array
  - integer (i4), intent(in) :: lenBLOCKS
  - integer (i4), intent(in) :: compDOF( ! Global degrees of freedom for comp decomposition
  - integer (i4), intent(in) :: ioDofR( ! Global degrees of freedom for I/O decomp (Read op)
  - integer (i4), intent(in) :: ioDofW( ! Global degrees of freedom for IO decomp (Write op)
  - integer (PIO_OFFSET), intent(in) :: start( , cnt( ! pNetCDF domain decomosition information
  - type (IO_desc_t), pointer, intent(out) :: IOdesc
    If Read and Write DOF arrays are the same only one need be passed.
    The start and count arguments are required only for (p)netcdf format, the will be ignored if passed with a binary file handle.
    PIO_initdecomp examples

    Although the IOsystem variable is not used directly after the open or create file calls it must remain in scope during all subsequent file operations. A single IOsystem may be used with several Files.
- integer function PIO_OpenFile(IOsystem,File,iotype, fname, mode)
- integer function PIO_CreateFile(IOsystem, File, iotype, fname, mode)
  - type (IOSystem_desc_t), intent(inout) :: iosystem
  - type (File_desc_t), intent(out) :: File
  - integer, intent(in) :: iotype
    - iotype_netcdf, iotype_pnetcdf, iotype_bin
  - character(len=☆), intent(in) :: fname
  - integer,intent(in), optional :: mode
    - mode can be used to pass mode options to (p)netcdf
    - default mode for open call is pio_nowrite
    - default mode for create call is pio_clobber, pio_nofill, 64bit_offset

- subroutine PIO_CloseFile(File)
  - type (File_desc_t),intent(inout) :: File
- subroutine PIO_write_darray(data_file,varDesc,IOdesc, array,iostat, fillval)
  - type (File_desc_t), intent(inout) :: data_file ! file information
  - type (IOsystem_desc_t), intent(inout) :: iosystem ! io subsystem information
  - type (var_desc_t), intent(inout) :: varDesc ! varable descriptor
  - type (io_desc_t), intent(inout) :: iodesc ! io descripter defined in initdecomp
  - intent(in) :: array ! array to be written (currently integer, real*4 and real8 types are supported, 1 dimension
  - integer, intent(out) :: iostat ! error return code
  - intent(in), optional :: fillvalue ! same type as array, a fillvalue for pio to use in the case of missing data
- subroutine PIO_read_darray(data_file,varDesc,iodesc,array,iostat)
  - type (File_desc_t), intent(inout) :: data_file ! info about data file
  - type (var_desc_t), intent(inout) :: varDesc ! variable descriptor
  - type (io_desc_t), intent(inout) :: iosystem
  - intent(in) :: array ! array to be read currently integer, real*4 and real8 types are supported, 1 dimension)
  - integer, intent(out) :: iostat ! error return code

- subroutine PIO_SetDebugLevel(level)

    - integer(i4), intent(in) :: level

      Prints debug information from the library, level can be 0 to 4 (verbose).
- subroutine PIO_SetFrame(VarDesc, frame)
    - type (Var_desc_t), intent(in) :: varDesc
    - integer(i4) :: frame

      Set the (p)netcdf unlimited dimension pointer for the variable described by VarDesc to record frame.
- subroutine PIO_AdvanceFrame(Vardesc)
    - type (Var_desc_t), intent(in) :: varDesc

      Advance the (p)netcdf unlimited dimension pointer for the variable described by VarDesc by adding 1.

      ### currently only implimented in (p)netcdf
- subroutine PIO_SetErrorHandling(File, method) subroutine PIO_SetErrorHandling(IOsystem, method)
    - type(file_desc_t), intent(inout) :: file
    - integer, intent(in) :: method
        - PIO_INTERNAL_ERROR (default): handle errors internally, print a message and abort on error from any task
        - PIO_BCAST_ERROR: broadcast an error from IO rank 0 and return on all tasks
        - PIO_RETURN_ERROR: do nothing, return error values

      ### Interfaces to mirror netCDF/pNetCDF
- integer function PIO_put_vara(File,varid, start, count, ival)
- integer function PIO_put_var1(File,varid, index, ival)
- integer function PIO_put_var(File,varid,ival)
    - type(File_Desc_t), intent(in) :: File
    - integer, intent(in) :: varid
    - integer, intent(in) :: index
    - integer, intent(in) :: start
    - integer, intent(in) :: count
    - Unknown macro: {character(len=☆) | integer(i4) | real(r4) | real(r8)}

      , intent(in) :: ival(🙂

      These functions writes a variable from IO node 0 to the file. They must be called collectively.

        - integer function PIO_def_var(File,name,type,dimids,varDesc) result(ierr)
            - type (File_desc_t), intent(in) :: File
            - character(len=☆), intent(in) :: name
            - integer, intent(in) :: type
            - integer, intent(in) :: dimids(🙂
            - type (Var_desc_t), intent(inout) :: varDesc
        - integer function PIO_inq_varid(File,name,varDesc) result(ierr)
            - type (File_desc_t), intent(in) :: File
            - character(len=☆), intent(in) :: name
            - type (Var_desc_t), intent(inout) :: varDesc
        - integer function PIO_EndDef(File) result(ierr)
            - type (File_desc_t), intent(inout) :: File

              ### Note to developers: The EndDef and Redef functions are known to be expensive, it is recommended to avoid redef and call enddef only once per file if possible.
        - integer function PIO_ReDef(File) result(ierr)
            - type (File_desc_t), intent(inout) :: File
        - integer function PIO_get_att(File,varid,name,value)
            - type (File_desc_t), intent(in) :: File
            - integer(i4), intent(in) :: varid
            - character(len=☆), intent(in) :: name

      , intent(out) :: value

- integer function PIO_put_att(File,varid,name,value)
    - type (File_desc_t), intent(in) :: File
    - integer(i4), intent(in) :: varid
    - character(len=☆), intent(in) :: name
    - Unknown macro: {character(len=☆) | integer(i4) | real(r4) | real(r8)}

      , intent(in) :: value

- integer function PIO_inquire(File,nDimensions,nVariables,nAttributes,unlimitedDimID)
    - type (File_desc_t), intent(in) :: File
    - integer, optional, intent(out) :: nDimensions ! number of dimensions
    - integer, optional, intent(out) :: nVariables ! number of variables
    - integer, optional, intent(out) :: nAttributes ! number of global attributes
    - integer, optinal, intent(out) :: unlimitedDimID ! ID of unlimited dimension
- integer function PIO_inq_attname(File,varid,attnum,name)
    - type (File_desc_t), intent(inout) :: File
    - integer, intent(out) :: varid !Variable ID
    - integer, intent(out) :: attnum ! attribute number
    - character(len=), intent(out) :: name
- integer function PIO_inq_dimid(File,name,dimid)
    - type (File_desc_t), intent(in) :: File
    - character(len=☆), intent(in) :: name

    - integer, intent(out) :: dimid !dimension ID
- integer function PIO_def_dim(File,name,len,dimid)
    - type (File_desc_t), intent(in) :: File
    - character(len=☆), intent(in) :: name
    - integer(i4), intent(in) :: len
    - integer(i4), intent(out) :: dimid
- integer function inq_att(File,varid,name,xtype,len)
- integer function inq_attname(File,varid,attnum,name)
- integer function inq_varid(File,name,varDesc)
- integer function inq_varname(File,varDesc,name)
- integer function inq_vartype(File,varDesc,xtype)
- integer function inq_varndims(File,varDesc,ndims)
- integer function inq_varnatts(File,varDesc,natts)
- integer function inq_dimid(File,name,dimid)
- integer function inq_dimname(File,dimid,dimname)
- integer function inq_dimlen(File,dimid,dimlen)
- integer function def_dim(File,name,len,dimid)
- integer function copy_att(infile, invarid, name, outfile, outvarid)