# HDF5 Fortran User Notes

## About the source code organization

The Fortran APIs are organized in modules parallel to the HDF5 Interfaces. Each module is in a separate file with the name `H5*ff.f`. Corresponding C stubs are in the `H5*f.c` files. For example, the Fortran File APIs are in the file `H5Fff.f` and the corresponding C stubs are in the file `H5Ff.c`.

Each module contains Fortran definitions of the constants, interfaces to the subroutines if needed, and the subroutines themselves.

Users must use constant names in their programs instead of the numerical values, as the numerical values are subject to change without notice.

## About the Fortran APIs

The Fortran APIs come in the form of Fortran subroutines with the following characteristics:

- Each Fortran subroutine name is derived from the corresponding C function name by adding `"_f"` to the name. For example, the name of the C function to create an HDF5 file is `H5Fcreate`; the corresponding Fortran subroutine is `h5fcreate_f`.

- A description of each implemented Fortran subroutine and its parameters can be found following the description of the corresponding C function in the HDF5 Reference Manual provided with this release.

- The parameter list for each Fortran subroutine has two more parameters than the corresponding C function. These additional parameters hold the return value and an error code. The order of the Fortran subroutine parameters may differ from the order of the C function parameters. The Fortran subroutine parameters are listed in the following order:
  - Required input parameters
  - Output parameters, including return value and error code
  - Optional input parameters

  For example, the C function to create a dataset has the following prototype:

  ```
  hid_t H5Dcreate(hid_it loc_id, char *name, hid_t type_id,
               hid_t space_id, hid_t creation_prp);
  ```

  The corresponding Fortran subroutine has the following form:

  ```
  SUBROUTINE h5dcreate_f(loc_id, name, type_id, space_id, dset_id,
               hdferr, creation_prp)
  ```

  The first four parameters of the Fortran subroutine correspond to the C function parameters. The fifth parameter, `dset_id`, is an output parameter and contains a valid dataset identifier if the value of the sixth output parameter hdferr indicates successful completion. (Error code descriptions are provided with the subroutine descriptions in the Reference Manual.) The seventh input parameter, `creation_prp`, is optional, and may be omitted when the default creation property list is used.
- Parameters to the Fortran subroutines have one of the following predefined datatypes (see the file `H5fortran_types.f90` for `KIND` def initions):
  - *INTEGER(HID_T)* compares with the `hid_t` datatype in the HDF5 C APIs.
  - `INTEGER(HSIZE_T)` compares with `hsize_t` in the HDF5 C APIs.
  - `INTEGER(HSSIZE_T)` compares with `hssize_t` in the HDF5 C APIs.
  - `INTEGER(SIZE_T)` compares with the C `size_t` datatype.

  These integer types usually correspond to 4 or 8 byte integers, depending on the FORTRAN90 compiler and the corresponding HDF5 C library definitions.

  The H5R module defines two types of references:
  - `TYPE(HOBJ_REF_T_F)` compares to `hobj_ref_t` in the HDF5 C API.
  - `TYPE(HDSET_REG_REF_T_F)` compares to `hdset_reg_ref_t` in the HDF5 C API.
- Each Fortran application must call the `h5open_f` subroutine to initialize the Fortran interface before calling any HDF5 Fortran subroutine. It is a good practice to call the `h5close_f` subroutine after all calls to the HDF5 Fortran library to close the Fortran interface.

- List of the predefined datatypes can be found in the HDF5 Reference Manual provided with this release. See HDF5 Predefined Datatypes .

- When a C application reads data stored from a Fortran program, the data will appear to be transposed due to the difference in the C and Fortran storage orders. For example, if Fortran writes a 4x6 two-dimensional dataset to the file, a C program will read it as a 6x4 two-dimensional dataset into memory. The HDF5 C utilities h5dump and h5ls will also display transposed data, if data is written from a

Fortran program.

- Fortran indices are 1-based.
- Compound datatype datasets can be written or read by atomic fields only.