

---

# HDF5 Fortran Guide

---

---

Release 1.4.1  
April, 2001

Hierarchical Data Format (HDF) Group  
National Center for Supercomputing Applications (NCSA)  
University of Illinois at Urbana-Champaign (UIUC)

---

**Includes:**

- *HDF5 Fortran90 User's Notes*
  - *HDF5 Fortran90 Reference Manual*
  - *HDF5 Fortran90 Flags and Datatypes*
-

## Copyright Notice and Statement for NCSA HDF5 (Hierarchical Data Format 5) Software Library and Utilities

NCSA HDF5 (Hierarchical Data Format 5) Software Library and Utilities  
Copyright 1998, 1999, 2000, 2001 by the Board of Trustees of the University of Illinois  
**All rights reserved.**

Contributors: National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign (UIUC), Lawrence Livermore National Laboratory (LLNL), Sandia National Laboratories (SNL), Los Alamos National Laboratory (LANL), Jean-loup Gailly and Mark Adler (gzip library).

1. Redistribution and use in source and binary forms, with or without modification, are permitted for any purpose (including commercial purposes) provided that the following conditions are met:
2. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
3. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation and/or materials provided with the distribution.
4. In addition, redistributions of modified forms of the source or binary code must carry prominent notices stating that the original code was changed and the date of the change.
5. All publications or advertising materials mentioning features or use of this software are asked, but not required, to acknowledge that it was developed by the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign and to credit the contributors.
6. Neither the name of the University nor the names of the Contributors may be used to endorse or promote products derived from this software without specific prior written permission from the University or the Contributors.
7. **THIS SOFTWARE IS PROVIDED BY THE UNIVERSITY AND THE CONTRIBUTORS "AS IS" WITH NO WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED.** In no event shall the University or the Contributors be liable for any damages suffered by the users arising out of the use of this software, even if advised of the possibility of such damage.

---

Portions of HDF5 were developed with support from the University of California, Lawrence Livermore National Laboratory (UC LLNL). The following statement applies to those portions of the product and must be retained in any redistribution of source code, binaries, documentation, and/or accompanying materials:

This work was partially produced at the University of California, Lawrence Livermore National Laboratory (UC LLNL) under contract no. W-7405-ENG-48 (Contract 48) between the U.S. Department of Energy (DOE) and The Regents of the University of California (University) for the operation of UC LLNL.

**DISCLAIMER:** This work was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately- owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

## Table of Contents

<b>HDF5 Fortran90 User's Notes .....</b>	<b>1</b>
About the source code organization .....	1
About the Fortran APIs .....	1
<b>HDF5 Fortran90 Reference Manual .....</b>	<b>3</b>
The FORTRAN 90 API to HDF5 h5: General Library .....	3
The FORTRAN 90 API to HDF5 h5a: Attributes .....	5
The FORTRAN 90 API to HDF5 h5d: Datasets .....	9
The FORTRAN 90 API to HDF5 h5d: Datasets .....	15
The FORTRAN 90 API to HDF5 h5e: Error Handling .....	21
The FORTRAN 90 API to HDF5 h5f: Files .....	23
The FORTRAN 90 API to HDF5 h5g: Groups .....	25
The FORTRAN 90 API to HDF5 h5i: Identifiers .....	29
The FORTRAN 90 API to HDF5 h5p: Property Lists .....	31
The FORTRAN 90 API to HDF5 h5r: References .....	47
The FORTRAN 90 API to HDF5 h5s: Dataspaces .....	49
The FORTRAN 90 API to HDF5 h5t: Datatypes .....	57
<b>HDF5 Fortran90 Flags and Datatypes .....</b>	<b>73</b>
Fortran90 Datatypes .....	73
Fortran90 Flags .....	73



# HDF5 Fortran90 User's 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.
- 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, and
- optional input parameters.

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

```
hid_t H5Dcreate(hid_t 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 definitions):

<code>INTEGER(HID_T)</code>	compares with <code>hid_t</code> type in HDF5 C APIs
<code>INTEGER(HSIZE_T)</code>	compares with <code>hsize_t</code> in HDF5 C APIs
<code>INTEGER(HSSIZE_T)</code>	compares with <code>hssize_t</code> in HDF5 C APIs
<code>INTEGER(SIZE_T)</code>	compares with the C <code>size_t</code> type

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:

<code>TYPE(HOBJ_REF_T_F)</code>	compares to <code>hobj_ref_t</code> in HDF5 C API
<code>TYPE(HDSET_REG_REF_T_F)</code>	compares to <code>hdset_reg_ref_t</code> in HDF5 C API

- Each Fortran application must call the `h5open_f` subroutine to initialize the Fortran interface and the HDF5 C Library before calling any HDF5 Fortran subroutine. The application must call the `h5close_f` subroutine after all calls to the HDF5 Fortran Library to close the Fortran interface and HDF5 C Library.
  - List of the predefined datatypes can be found in the HDF5 Reference Manual provided with this release. See "HDF5 Predefined Datatypes" in the *HDF5 Reference Manual*.
  - 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.
-

# HDF5 Fortran90 Reference Manual

## The FORTRAN 90 API to HDF5 h5: General Library

---

### **FORTRAN interface: h5open\_f**

**Purpose:** Initializes the HDF5 library and the Fortran90 interface.

```
SUBROUTINE h5open_f(hdferr)
  IMPLICIT NONE
  INTEGER, INTENT(OUT) :: hdferr      !Error code

END SUBROUTINE h5open_f
```

---

### **FORTRAN interface: h5close\_f**

**Purpose:** Closes the HDF5 library and the Fortran90 interface.

```
SUBROUTINE h5close_f(hdferr)
  IMPLICIT NONE
  INTEGER, INTENT(OUT) :: hdferr      !Error code

END SUBROUTINE h5close_f
```

---



# The FORTRAN 90 API to HDF5

## h5a: Attributes

### FORTRAN interface: h5aclose\_f

```

SUBROUTINE h5aclose_f(attr_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(OUT) :: attr_id  !Attribute identifier
  INTEGER, INTENT(OUT) :: hdferr         !Error code
                                          !0 on success and -1 on failure
END SUBROUTINE h5aclose_f

```

### FORTRAN interface: h5acreate\_f

```

SUBROUTINE h5acreate_f(obj_id, name, type_id, space_id, attr_id, &
                      hdferr, creation_prp)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: obj_id    !Object identifier
  CHARACTER(LEN=*), INTENT(IN) :: name    !Attribute name
  INTEGER(HID_T), INTENT(IN) :: type_id   !Attribute datatype identifier
  INTEGER(HID_T), INTENT(IN) :: space_id  !Attribute dataspace identifier
  INTEGER(HID_T), INTENT(OUT) :: attr_id  !Attribute identifier
  INTEGER, INTENT(OUT) :: hdferr         !Error code:
                                          !0 on success and -1 on failure
  INTEGER(HID_T), OPTIONAL, INTENT(IN) :: creation_prp
                                          !Attribute creation property
                                          !list identifier
END SUBROUTINE h5acreate_f

```

### FORTRAN interface: h5adelete\_f

```

SUBROUTINE h5adelete_f(obj_id, name, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: obj_id    !Object identifier
  CHARACTER(LEN=*), INTENT(IN) :: name    !Attribute name
  INTEGER, INTENT(OUT) :: hdferr         !Error code

```

```
!0 on success and -1 on failure  
END SUBROUTINE h5adelete_f
```

---

**FORTRAN interface: h5aget\_name\_f**

```
SUBROUTINE h5aget_name_f(attr_id, size, buf, hdferr)  
  IMPLICIT NONE  
  INTEGER(HID_T), INTENT(IN) :: attr_id !Attribute identifier  
  INTEGER, INTENT(IN) :: size !Buffer size  
  CHARACTER(LEN=*), INTENT(OUT) :: buf !Buffer to hold attribute name  
  INTEGER, INTENT(OUT) :: hdferr !Error code : name length  
                                     !on success and -1 on failure  
END SUBROUTINE h5aget_name_f
```

---

**FORTRAN interface: h5aget\_num\_attrs\_f**

```
SUBROUTINE h5aget_num_attrs_f(obj_id, attr_num, hdferr)  
  IMPLICIT NONE  
  INTEGER(HID_T), INTENT(IN) :: obj_id !Object identifier  
  INTEGER, INTENT(OUT) :: attr_num !Number of attributes of the  
                                     !object  
  INTEGER, INTENT(OUT) :: hdferr !Error code  
                                     !0 on success and -1 on failure  
END SUBROUTINE h5aget_num_attrs_f
```

---

**FORTRAN interface: h5aget\_space\_f**

```
SUBROUTINE h5aget_space_f(attr_id, space_id, hdferr)  
  IMPLICIT NONE  
  INTEGER(HID_T), INTENT(IN) :: attr_id !Attribute identifier  
  INTEGER(HID_T), INTENT(OUT) :: space_id !Attribute dataspace identifier  
  INTEGER, INTENT(OUT) :: hdferr !Error code  
                                     !0 on success and -1 on failure  
END SUBROUTINE h5aget_space_f
```

---

**FORTTRAN interface: h5aget\_type\_f**

```
SUBROUTINE h5aget_type_f(attr_id, type_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: attr_id !Attribute identifier
  INTEGER(HID_T), INTENT(OUT) :: type_id !Attribute datatype identifier
  INTEGER, INTENT(OUT) :: hdferr          !Error code
                                          !0 on success and -1 on failure
END SUBROUTINE h5aget_type_f
```

---

**FORTTRAN interface: h5aopen\_idx\_f**

```
SUBROUTINE h5aopen_idx_f(obj_id, index, attr_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: obj_id    !Object identifier
  INTEGER, INTENT(IN) :: index           !Attribute index
  INTEGER(HID_T), INTENT(OUT) :: attr_id  !Attribute identifier
  INTEGER, INTENT(OUT) :: hdferr         !Error code
                                          !0 on success and -1 on failure
END SUBROUTINE h5aopen_idx_f
```

---

**FORTTRAN interface: h5aopen\_name\_f**

```
SUBROUTINE h5aopen_name_f(obj_id, name, attr_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: obj_id    !Object identifier
  CHARACTER(LEN=*), INTENT(IN) :: name    !Attribute name
  INTEGER(HID_T), INTENT(OUT) :: attr_id  !Attribute identifier
  INTEGER, INTENT(OUT) :: hdferr         !Error code
                                          !0 on success and -1 on failure
END SUBROUTINE h5aopen_name_f
```

---

**FORTRAN interface: h5aread\_f**

```
SUBROUTINE h5aread_f(attr_id, memtype_id, buf, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: attr_id      !Attribute identifier
  INTEGER(HID_T), INTENT(IN) :: memtype_id  !Attribute datatype
                                          !identifier (in memory)
  TYPE(VOID), INTENT(OUT) :: buf           !Attribute data
  INTEGER, INTENT(OUT) :: hdferr           !Error code
                                          !0 on success and -1 on failure
END SUBROUTINE h5aread_f
```

---

**FORTRAN interface: h5awrite\_f**

```
SUBROUTINE h5awrite_f(attr_id, memtype_id, buf, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: attr_id      !Attribute identifier
  INTEGER(HID_T), INTENT(IN) :: memtype_id  !Attribute datatype
                                          !identifier (in memory)
  TYPE(VOID), INTENT(IN) :: buf             !Attribute data
  INTEGER, INTENT(OUT) :: hdferr           !Error code
                                          !0 on success and -1 on failure
END SUBROUTINE h5awrite_f
```

---

---

# The FORTRAN 90 API to HDF5

## h5d: Datasets

---

**FORTRAN interface: h5dclose\_f**

```
SUBROUTINE h5dclose_f(dset_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: dset_id !Dataset identifier
  INTEGER, INTENT(OUT) :: hdferr       !Error code
                                       !0 on success and -1 on failure
END SUBROUTINE h5dclose_f
```

---

**FORTRAN interface: h5dcreate\_f**

```
SUBROUTINE h5dcreate_f(loc_id, name, type_id, space_id, dset_id, &
                      hdferr, creation_prp)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: loc_id !File or group identifier
  CHARACTER(LEN=*), INTENT(IN) :: name !Name of the dataset
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER(HID_T), INTENT(IN) :: space_id !Dataspace identifier
  INTEGER(HID_T), INTENT(OUT) :: dset_id !Dataset identifier
  INTEGER, INTENT(OUT) :: hdferr       !Error code
                                       !0 on success and -1 on failure
  INTEGER(HID_T), OPTIONAL, INTENT(IN) :: creation_prp
                                       !Dataset creation property
                                       !list identifier , default
                                       !value is H5P_DEFAULT_F (6)
END SUBROUTINE h5dcreate_f
```

---

**FORTRAN interface: h5dextend\_f**

```
SUBROUTINE h5dextend_f(dataset_id, size, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: dataset_id !Dataset identifier
  INTEGER(HSIZE_T), DIMENSION(*), INTENT(IN) :: size
                                       !Array containing
```

---

```
                                !dimensions' sizes
    INTEGER, INTENT(OUT) :: hdferr      !Error code
                                        !0 on success and -1 on failure
END SUBROUTINE h5dextend_f
```

---

**FORTRAN interface: h5dget\_create\_plist\_f**

```
SUBROUTINE h5dget_create_plist_f(dataset_id, creation_prp, hdferr)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: dataset_id      !Dataset identifier
    INTEGER(HID_T), INTENT(OUT) :: creation_id    !Dataset creation
                                                !property list identifier
    INTEGER, INTENT(OUT) :: hdferr                !Error code
                                                !0 on success and -1 on failure
END SUBROUTINE h5dget_create_plist_f
```

---

**FORTRAN interface: h5dget\_space\_f**

```
SUBROUTINE h5dget_space_f(dataset_id, dataspace_id, hdferr)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: dataset_id      !Dataset identifier
    INTEGER(HID_T), INTENT(OUT) :: dataspace_id   !Dataspace identifier
    INTEGER, INTENT(OUT) :: hdferr                !Error code
                                                !0 on success and -1 on failure
END SUBROUTINE h5dget_space_f
```

---

**FORTRAN interface: h5dget\_type\_f**

```
SUBROUTINE h5dget_type_f(dataset_id, datatype_id, hdferr)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: dataset_id      !Dataset identifier
    INTEGER(HID_T), INTENT(OUT) :: datatype_id    !Datatype identifier
    INTEGER, INTENT(OUT) :: hdferr                !Error code
                                                !0 on success and -1 on failure
END SUBROUTINE h5dget_type_f
```

---

**FORTRAN interface: h5dopen\_f**

```

SUBROUTINE h5dopen_f(loc_id, name, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: loc_id      !File or group identifier
  CHARACTER(LEN=*), INTENT(IN) :: name     !Name of the dataset
  INTEGER (HID_T), INTENT (OUT) :: dset_id !Dataset identifier
  INTEGER, INTENT(OUT) :: hdferr           !Error code
                                           !0 on success and -1 on failure
END SUBROUTINE h5dopen_f

```

**FORTRAN interface: h5dread\_f** (for all datatypes except object and dataset region references)

```

SUBROUTINE h5dread_f(dset_id, mem_type_id, buf, hdferr, &
                    mem_space_id, file_space_id, xfer_prp)

  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: dset_id      !Dataset identifier
  INTEGER(HID_T), INTENT(IN) :: mem_type_id !Memory datatype identifier
  TYPE, INTENT(IN) :: buf(*,...*)          !Data buffer of rank k
  INTEGER, INTENT(OUT) :: hdferr            !Error code
                                           !0 on success and -1 on failure

  INTEGER(HID_T), OPTIONAL, INTENT(IN) :: mem_space_id
                                           !Memory dataspace identifier
                                           !Default value is H5S_ALL_F

  INTEGER(HID_T), OPTIONAL, INTENT(IN) :: file_space_id
                                           !File dataspace identifier
                                           !Default value is H5S_ALL_F

  INTEGER(HID_T), OPTIONAL, INTENT(IN) :: xfer_prp
                                           !Transfer property list identifier
                                           !Default value is H5P_DEFAULT_F

END SUBROUTINE h5dread_f

```

**FORTRAN interface: h5dread\_f** (for object reference and dataset region reference datatypes)

```

SUBROUTINE h5dread_f(dset_id, mem_type_id, buf, n, hdferr, &
                    mem_space_id, file_space_id, xfer_prp)

  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: dset_id      !Dataset identifier
  INTEGER(HID_T), INTENT(IN) :: mem_type_id !Memory datatype identifier
  TYPE, INTENT(IN) :: buf(*)                !Data buffer of rank 1

```

```
TYPE, INTENT(IN) :: n                !Buffer size, 1 dimension only
INTEGER, INTENT(OUT) :: hdferr        !Error code
                                      !0 on success and -1 on failure
INTEGER(HID_T), OPTIONAL, INTENT(IN) :: mem_space_id
                                      !Memory dataspace identifier
                                      !Default value is H5S_ALL_F
INTEGER(HID_T), OPTIONAL, INTENT(IN) :: file_space_id
                                      !File dataspace identifier
                                      !Default value is H5S_ALL_F
INTEGER(HID_T), OPTIONAL, INTENT(IN) :: xfer_prp
                                      !Transfer property list identifier
                                      !Default value is H5P_DEFAULT_F

END SUBROUTINE h5dread_f
```

---

**FORTRAN interface: h5dwrite\_f** (for all datatypes except object and dataset region references)

```
SUBROUTINE h5dwrite_f(dset_id, mem_type_id, buf, hdferr, &
                    mem_space_id, file_space_id, xfer_prp)

IMPLICIT NONE
INTEGER(HID_T), INTENT(IN) :: dset_id    !Dataset identifier
INTEGER(HID_T), INTENT(IN) :: mem_type_id !Memory datatype identifier
TYPE, INTENT(IN) :: buf(*,...*)        !Data buffer of rank k
INTEGER, INTENT(OUT) :: hdferr          !Error code
                                      !0 on success and -1 on failure
INTEGER(HID_T), OPTIONAL, INTENT(IN) :: mem_space_id
                                      !Memory dataspace identifier
                                      !Default value is H5S_ALL_F
INTEGER(HID_T), OPTIONAL, INTENT(IN) :: file_space_id
                                      !File dataspace identifier
                                      !Default value is H5S_ALL_F
INTEGER(HID_T), OPTIONAL, INTENT(IN) :: xfer_prp
                                      !Transfer property list identifier
                                      !Default value is H5P_DEFAULT_F

END SUBROUTINE h5dwrite_f
```

**FORTRAN interface: h5dwrite\_f** (for object reference and dataset region reference datatypes)

```
SUBROUTINE h5dwrite_f(dset_id, mem_type_id, buf, n, hdferr, &
                    mem_space_id, file_space_id, xfer_prp)

  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: dset_id      !Dataset identifier
  INTEGER(HID_T), INTENT(IN) :: mem_type_id !Memory datatype identifier
  TYPE, INTENT(IN) :: buf(*)                !Data buffer of rank 1
  TYPE, INTENT(IN) :: n                     !Buffer size, 1 dimension only
  INTEGER, INTENT(OUT) :: hdferr            !Error code
                                           !0 on success and -1 on failure
  INTEGER(HID_T), OPTIONAL, INTENT(IN) :: mem_space_id
                                           !Memory dataspace identifier
                                           !Default value is H5S_ALL_F
  INTEGER(HID_T), OPTIONAL, INTENT(IN) :: file_space_id
                                           !File dataspace identifier
                                           !Default value is H5S_ALL_F
  INTEGER(HID_T), OPTIONAL, INTENT(IN) :: xfer_prp
                                           !Transfer property list identifier
                                           !Default value is H5P_DEFAULT_F

END SUBROUTINE h5dwrite_f
```



# The FORTRAN 90 API to HDF5

## h5d: Datasets

---

### FORTRAN interface: h5dclose\_f

```

SUBROUTINE h5dclose_f(dset_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: dset_id  !Dataset identifier
  INTEGER, INTENT(OUT) :: hdferr        !Error code
                                          !0 on success and -1 on failure
END SUBROUTINE h5dclose_f

```

---

### FORTRAN interface: h5dcreate\_f

```

SUBROUTINE h5dcreate_f(loc_id, name, type_id, space_id, dset_id, &
                      hdferr, creation_prp)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: loc_id    !File or group identifier
  CHARACTER(LEN=*), INTENT(IN) :: name    !Name of the dataset
  INTEGER(HID_T), INTENT(IN) :: type_id  !Datatype identifier
  INTEGER(HID_T), INTENT(IN) :: space_id !Dataspace identifier
  INTEGER(HID_T), INTENT(OUT) :: dset_id !Dataset identifier
  INTEGER, INTENT(OUT) :: hdferr         !Error code
                                          !0 on success and -1 on failure
  INTEGER(HID_T), OPTIONAL, INTENT(IN) :: creation_prp
                                          !Dataset creation property
                                          !list identifier , default
                                          !value is H5P_DEFAULT_F (6)
END SUBROUTINE h5dcreate_f

```

---

### FORTRAN interface: h5dextend\_f

```

SUBROUTINE h5dextend_f(dataset_id, size, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: dataset_id  !Dataset identifier
  INTEGER(HSIZE_T), DIMENSION(*), INTENT(IN) :: size
                                          !Array containing

```

---

```
                                !dimensions' sizes
    INTEGER, INTENT(OUT) :: hdferr      !Error code
                                        !0 on success and -1 on failure
END SUBROUTINE h5dextend_f
```

---

**FORTRAN interface: h5dget\_create\_plist\_f**

```
SUBROUTINE h5dget_create_plist_f(dataset_id, creation_prp, hdferr)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: dataset_id      !Dataset identifier
    INTEGER(HID_T), INTENT(OUT) :: creation_id    !Dataset creation
                                                !property list identifier
    INTEGER, INTENT(OUT) :: hdferr              !Error code
                                                !0 on success and -1 on failure
END SUBROUTINE h5dget_create_plist_f
```

---

**FORTRAN interface: h5dget\_space\_f**

```
SUBROUTINE h5dget_space_f(dataset_id, dataspace_id, hdferr)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: dataset_id      !Dataset identifier
    INTEGER(HID_T), INTENT(OUT) :: dataspace_id  !Dataspace identifier
    INTEGER, INTENT(OUT) :: hdferr              !Error code
                                                !0 on success and -1 on failure
END SUBROUTINE h5dget_space_f
```

---

**FORTRAN interface: h5dget\_type\_f**

```
SUBROUTINE h5dget_type_f(dataset_id, datatype_id, hdferr)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: dataset_id      !Dataset identifier
    INTEGER(HID_T), INTENT(OUT) :: datatype_id   !Datatype identifier
    INTEGER, INTENT(OUT) :: hdferr              !Error code
                                                !0 on success and -1 on failure
END SUBROUTINE h5dget_type_f
```

**FORTRAN interface: h5dopen\_f**

```

SUBROUTINE h5dopen_f(loc_id, name, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: loc_id      !File or group identifier
  CHARACTER(LEN=*), INTENT(IN) :: name     !Name of the dataset
  INTEGER, INTENT(OUT) :: hdferr           !Error code
                                           !0 on success and -1 on failure
END SUBROUTINE h5dopen_f

```

**FORTRAN interface: h5dread\_f** (for all datatypes except object and dataset region references)

```

SUBROUTINE h5dread_f(dset_id, mem_type_id, buf, hdferr, &
                    mem_space_id, file_space_id, xfer_prp)

  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: dset_id      !Dataset identifier
  INTEGER(HID_T), INTENT(IN) :: mem_type_id !Memory datatype identifier
  TYPE, INTENT(IN) :: buf(*,...*)          !Data buffer of rank k
  INTEGER, INTENT(OUT) :: hdferr            !Error code
                                           !0 on success and -1 on failure
  INTEGER(HID_T), OPTIONAL, INTENT(IN) :: mem_space_id
                                           !Memory dataspace identifier
                                           !Default value is H5S_ALL_F
  INTEGER(HID_T), OPTIONAL, INTENT(IN) :: file_space_id
                                           !File dataspace identifier
                                           !Default value is H5S_ALL_F
  INTEGER(HID_T), OPTIONAL, INTENT(IN) :: xfer_prp
                                           !Transfer property list identifier
                                           !Default value is H5P_DEFAULT_F

END SUBROUTINE h5dread_f

```

**FORTRAN interface: h5dread\_f** (for object reference and dataset region reference datatypes)

```
SUBROUTINE h5dread_f(dset_id, mem_type_id, buf, n, hdferr, &
                    mem_space_id, file_space_id, xfer_prp)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: dset_id    !Dataset identifier
    INTEGER(HID_T), INTENT(IN) :: mem_type_id !Memory datatype identifier
    TYPE, INTENT(IN) :: buf(*)              !Data buffer of rank 1
    TYPE, INTENT(IN) :: n                   !Buffer size, 1 dimension only
    INTEGER, INTENT(OUT) :: hdferr          !Error code
                                           !0 on success and -1 on failure
    INTEGER(HID_T), OPTIONAL, INTENT(IN) :: mem_space_id
                                           !Memory dataspace identifier
                                           !Default value is H5S_ALL_F
    INTEGER(HID_T), OPTIONAL, INTENT(IN) :: file_space_id
                                           !File dataspace identifier
                                           !Default value is H5S_ALL_F
    INTEGER(HID_T), OPTIONAL, INTENT(IN) :: xfer_prp
                                           !Transfer property list identifier
                                           !Default value is H5P_DEFAULT_F

END SUBROUTINE h5dread_f
```

---

**FORTRAN interface: h5dwrite\_f** (for all datatypes except object and dataset region references)

```
SUBROUTINE h5dwrite_f(dset_id, mem_type_id, buf, hdferr, &
                    mem_space_id, file_space_id, xfer_prp)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: dset_id    !Dataset identifier
    INTEGER(HID_T), INTENT(IN) :: mem_type_id !Memory datatype identifier
    TYPE, INTENT(IN) :: buf(*,...*)         !Data buffer of rank k
    INTEGER, INTENT(OUT) :: hdferr          !Error code
                                           !0 on success and -1 on failure
    INTEGER(HID_T), OPTIONAL, INTENT(IN) :: mem_space_id
                                           !Memory dataspace identifier
                                           !Default value is H5S_ALL_F
    INTEGER(HID_T), OPTIONAL, INTENT(IN) :: file_space_id
                                           !File dataspace identifier
                                           !Default value is H5S_ALL_F
```

```

INTEGER(HID_T), OPTIONAL, INTENT(IN) :: xfer_prp
                                !Transfer property list identifier
                                !Default value is H5P_DEFAULT_F

END SUBROUTINE h5dwrite_f

```

**FORTRAN interface: h5dwrite\_f** (for object reference and dataset region reference datatypes)

```

SUBROUTINE h5dwrite_f(dset_id, mem_type_id, buf, n, hdferr, &
                    mem_space_id, file_space_id, xfer_prp)

IMPLICIT NONE
INTEGER(HID_T), INTENT(IN) :: dset_id      !Dataset identifier
INTEGER(HID_T), INTENT(IN) :: mem_type_id !Memory datatype identifier
TYPE, INTENT(IN) :: buf(*)                !Data buffer of rank 1
TYPE, INTENT(IN) :: n                    !Buffer size, 1 dimension only
INTEGER, INTENT(OUT) :: hdferr            !Error code
                                        !0 on success and -1 on failure

INTEGER(HID_T), OPTIONAL, INTENT(IN) :: mem_space_id
                                        !Memory dataspace identifier
                                        ! Default value is H5S_ALL_F

INTEGER(HID_T), OPTIONAL, INTENT(IN) :: file_space_id
                                        !File dataspace identifier
                                        !Default value is H5S_ALL_F

INTEGER(HID_T), OPTIONAL, INTENT(IN) :: xfer_prp
                                        !Transfer property list identifier
                                        !Default value is H5P_DEFAULT_F

END SUBROUTINE h5dwrite_f

```



---

# The FORTRAN 90 API to HDF5

## h5e: Error Handling

---

**FORTRAN interface: h5eclear\_f**

```
SUBROUTINE h5eclear_f(hdferr)
  IMPLICIT NONE
  INTEGER, INTENT(OUT) :: hdferr !Error code

END SUBROUTINE h5eclear_f
```

---

**FORTRAN interface: h5eprint\_f**

```
SUBROUTINE h5eprint_f(hdferr, name)
  CHARACTER(LEN=*), OPTIONAL, INTENT(IN) :: name !File name
  INTEGER, INTENT(OUT) :: hdferr !Error code

END SUBROUTINE h5eprint_f
```

---

**FORTRAN interface: h5eget\_major\_f**

```
SUBROUTINE h5eget_major_f(error_no, name, hdferr)
  INTEGER, INTENT(IN) :: error_no !Major error number
  CHARACTER(LEN=*), INTENT(OUT) :: name !File name
  INTEGER, INTENT(OUT) :: hdferr !Error code

END SUBROUTINE h5eget_major_f
```

---

**FORTRAN interface: h5eget\_minor\_f**

```
SUBROUTINE h5eget_minor_f(error_no, name, hdferr)
  INTEGER, INTENT(IN) :: error_no !Major error number
  CHARACTER(LEN=*), INTENT(OUT) :: name !File name
  INTEGER, INTENT(OUT) :: hdferr !Error code

END SUBROUTINE h5eget_minor_f
```

---

**FORTRAN interface: h5set\_auto\_f**

```
SUBROUTINE h5set_auto_f(printflag, hdferr)
  INTEGER, INTENT(IN) :: printflag  !flag to turn automatic error
                                     !printing on or off
                                     !possible values are:
                                     !printon (1)
                                     !printoff(0)
  INTEGER, INTENT(OUT) :: hdferr    !Error code

END SUBROUTINE h5set_auto_f
```

---

## The FORTRAN 90 API to HDF5

### h5f: Files

---

#### **FORTRAN interface: h5fclose\_f**

```

SUBROUTINE h5fclose_f(file_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: file_id !File identifier
  INTEGER, INTENT(OUT) :: hdferr      !Error code
                                      !0 on success and -1 on failure
END SUBROUTINE h5fclose_f

```

---

#### **FORTRAN interface: h5fcreate\_f**

```

SUBROUTINE h5fcreate_f(name, access_flags, file_id, hdferr, &
                      creation_prp, access_prp)
  IMPLICIT NONE
  CHARACTER(LEN=*), INTENT(IN) :: name      !Name of the file
  INTEGER, INTENT(IN) :: access_flag       !File access flags
                                           !Possible values are:
                                           !H5F_ACC_RDWR_F
                                           !H5F_ACC_RDONLY_F
                                           !H5F_ACC_TRUNC_F
                                           !H5F_ACC_EXCL_F
                                           !H5F_ACC_DEBUG_F
  INTEGER(HID_T), INTENT(OUT) :: file_id !File identifier
  INTEGER, INTENT(OUT) :: hdferr         !Error code
                                           !0 on success and -1 on failure
  INTEGER(HID_T), OPTIONAL, INTENT(IN) :: creation_prp
                                           !File creation property
                                           !list identifier, if not
                                           !specified its value is
                                           !H5P_DEFAULT_F
  INTEGER(HID_T), OPTIONAL, INTENT(IN) :: access_prp
                                           !File access property list
                                           !identifier, if not
                                           ! specified its value is
                                           !H5P_DEFAULT_F

```

```
END SUBROUTINE h5fcreate_f
```

---

**FORTRAN interface: h5fis\_hdf5\_f**

```
SUBROUTINE h5fis_hdf5_f(name, status, hdferr)
  IMPLICIT NONE
  CHARACTER(LEN=*), INTENT(IN) :: name      !Name of the file
  LOGICAL, INTENT(OUT) :: status            !This parameter
                                              !indicates if file is
                                              !an HDF5 file
                                              !( TRUE or FALSE )
  INTEGER, INTENT(OUT) :: hdferr           !Error code
                                              !0 on success and -1 on failure
END SUBROUTINE h5fis_hdf5_f
```

---

**FORTRAN interface: h5fopen\_f**

```
SUBROUTINE h5fopen_f(name, access_flags, file_id, hdferr, &
                    access_prp)
  IMPLICIT NONE
  CHARACTER(LEN=*), INTENT(IN) :: name      !Name of the file
  INTEGER, INTENT(IN) :: access_flag        !File access flags
                                              !Possible values are:
                                              !H5F_ACC_RDWR_F
                                              !H5F_ACC_RDONLY_F
  INTEGER(HID_T), INTENT(OUT) :: file_id    !File identifier
  INTEGER, INTENT(OUT) :: hdferr           !Error code
                                              !0 on success and -1 on failure
  INTEGER(HID_T), OPTIONAL, INTENT(IN) :: access_prp
                                              !File access property list
                                              !identifier
END SUBROUTINE h5fopen_f
```

---

## The FORTRAN 90 API to HDF5

### h5g: Groups

#### FORTRAN interface: h5gclose\_f

```

SUBROUTINE h5gclose_f( gr_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: gr_id      !Group identifier
  INTEGER, INTENT(OUT) :: hdferr          !Error code
                                          !0 on success and -1 on failure
END SUBROUTINE h5gclose_f

```

#### FORTRAN interface: h5gcreate\_f

```

SUBROUTINE h5gcreate_f(loc_id, name, gr_id, hdferr, size_hint)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: loc_id    !File or group identifier
  CHARACTER(LEN=*), INTENT(IN) :: name    !Name of the group to be created
  INTEGER(HID_T), INTENT(OUT) :: gr_id    !Group identifier
  INTEGER, INTENT(OUT) :: hdferr         !Error code
                                          !0 on success and -1 on failure
  INTEGER(SIZE_T), OPTIONAL, INTENT(IN) :: size_hint
                                          !Number of bytes to store the names
                                          !of objects in the group.
                                          !Default value is OBJECT_NAMELEN_DEFAULT_F
END SUBROUTINE h5gcreate_f

```

#### FORTRAN interface: h5gget\_comment\_f

```

SUBROUTINE h5gget_comment_f(loc_id, name, size, buffer, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: loc_id    !File or group identifier
  CHARACTER(LEN=*), INTENT(IN) :: name    !Name of the object link
  CHARACTER(LEN=size), INTENT(OUT) :: buffer !Buffer to hold a
                                          !comment
  INTEGER, INTENT(OUT) :: hdferr          !Error code

```

```
                                !0 on success and -1 on failure
END SUBROUTINE h5gget_comment_f
```

---

**FORTRAN interface: h5gget\_linkval\_f**

```
SUBROUTINE h5gget_linkval_f(loc_id, name, size, buffer, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: loc_id    !File or group identifier
  CHARACTER(LEN=*), INTENT(IN) :: name    !Name of the symbolic link
  CHARACTER(LEN=size), INTENT(OUT) :: buffer !Buffer to hold a
                                           !name of the object
                                           !symbolic link points to
  INTEGER, INTENT(OUT) :: hdferr          !Error code
                                           !0 on success and -1 on failure
END SUBROUTINE h5gget_linkval_f
```

---

**FORTRAN interface: h5gget\_obj\_info\_idx\_f**

```
SUBROUTINE h5gget_obj_info_idx_f(loc_id, name, idx, &
                                obj_name, obj_type, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: loc_id    !File or group identifier
  CHARACTER(LEN=*), INTENT(IN) :: name    !Name of the group
  INTEGER, INTENT(IN) :: idx              !Index of member object
  CHARACTER(LEN=*), INTENT(OUT) :: obj_name !Name of the object
  INTEGER, INTENT(OUT) :: obj_type        !Object type :
                                           !H5G_LINK_F
                                           !H5G_GROUP_F
                                           !H5G_DATASET_F
                                           !H5G_TYPE_F
  INTEGER, INTENT(OUT) :: hdferr          !Error code
                                           !0 on success and -1 on failure
END SUBROUTINE h5gget_obj_info_idx_f
```

---

**FORTRAN interface: h5gmove\_f**

```
SUBROUTINE h5gmove_f(loc_id, name, new_name, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: loc_id    !File or group identifier
  CHARACTER(LEN=*), INTENT(IN) :: name    !Original name of an object
  CHARACTER(LEN=*), INTENT(IN) :: new_name !New name of an object
  INTEGER, INTENT(OUT) :: hdferr          !Error code
                                          !0 on success and -1 on failure
END SUBROUTINE h5gmove_f
```

---

**FORTRAN interface: h5gn\_members\_f**

```
SUBROUTINE h5gn_members_f(loc_id, name, nmembers, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: loc_id    !File or group identifier
  CHARACTER(LEN=*), INTENT(IN) :: name    !Name of the group
  INTEGER, INTENT(OUT) :: nmembers        !Number of members in the
                                          !group
  INTEGER, INTENT(OUT) :: hdferr          !Error code
                                          !0 on success and -1 on failure
END SUBROUTINE h5gn_members_f
```

---

**FORTRAN interface: h5gopen\_f**

```
SUBROUTINE h5gopen_f(loc_id, name, gr_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: loc_id    !File or group identifier
  CHARACTER(LEN=*), INTENT(IN) :: name    !Name of the group to open
  INTEGER(HID_T), INTENT(OUT) :: gr_id    !Group identifier
  INTEGER, INTENT(OUT) :: hdferr          !Error code
                                          !0 on success and -1 on failure
END SUBROUTINE h5gopen_f
```

---

**FORTRAN interface: h5gset\_comment\_f**

```
SUBROUTINE h5gset_comment_f(loc_id, name, comment, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: loc_id      !File or group identifier
  CHARACTER(LEN=*), INTENT(IN) :: name     !Name of object
  CHARACTER(LEN=*), INTENT(IN) :: comment  !Comment for the object
  INTEGER, INTENT(OUT) :: hdferr           !Error code
                                           !0 on success and -1 on failure
END SUBROUTINE h5gset_comment_f
```

---

**FORTRAN interface: h5gunlink\_f**

```
SUBROUTINE h5gunlink_f(loc_id, name, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: loc_id     !File or group identifier
  CHARACTER(LEN=*), INTENT(IN) :: name     !Name of the object to unlink
  INTEGER, INTENT(OUT) :: hdferr           !Error code
                                           !0 on success and -1 on failure
END SUBROUTINE h5gunlink_f
```

---

## The FORTRAN 90 API to HDF5

### h5i: Identifiers

---

**FORTRAN interface: h5iget\_type\_f**

```
SUBROUTINE h5iget_type_f(obj_id, type, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: obj_id  !Object identifier
  INTEGER, INTENT(OUT) :: type          !type of an object.
                                         !possible values are:
                                         !H5I_FILE_F(1)
                                         !H5I_GROUP_F(2)
                                         !H5I_DATATYPE_F(3)
                                         !H5I_DATASPACE_F(4)
                                         !H5I_DATASET_F(5)
                                         !H5I_ATTR_F(6)
                                         !H5I_BADID_F(-1)

  INTEGER, INTENT(OUT) :: hdferr        !Error code

END SUBROUTINE h5iget_type_f
```

---



# The FORTRAN 90 API to HDF5

## h5p: Property Lists

---

### FORTRAN interface: h5pclose\_f

```

SUBROUTINE h5pclose_f(prp_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id  !Property list identifier
  INTEGER, INTENT(OUT) :: hdferr       !Error code
                                       !0 on success and -1 on failure

END SUBROUTINE h5pclose_f

```

---

### FORTRAN interface: h5pcopy\_f

```

SUBROUTINE h5pcopy_f(prp_id, new_prp_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id  !Property list identifier
  INTEGER(HID_T), INTENT(OUT) :: new_prp_id
                                       !Identifier of property list
                                       !copy
  INTEGER, INTENT(OUT) :: hdferr       !Error code
                                       !0 on success and -1 on failure

END SUBROUTINE h5pcopy_f

```

---

### FORTRAN interface: h5pcreate\_f

```

SUBROUTINE h5pcreate_f(classtype, prp_id, hdferr)
  IMPLICIT NONE
  INTEGER, INTENT(IN) :: classtype  !The type of the property list
                                       !to be created. Possible values
                                       !are:
                                       !H5P_FILE_CREATE_F
                                       !H5P_FILE_ACCESS_F
                                       !H5P_DATASET_CREATE_F
                                       !H5P_DATASET_XFER_F

```

```
                                !H5P_MOUNT_F
INTEGER(HID_T), INTENT(OUT) :: prp_id !Property list identifier
INTEGER, INTENT(OUT) :: hdferr      !Error code
                                !0 on success and -1 on failure

END SUBROUTINE h5pcreate_f
```

---

**FORTRAN interface: h5pget\_chunk\_f**

```
SUBROUTINE h5pget_chunk_f(prp_id, ndims, dims, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(IN) :: ndims          !Number of chunk dimensions to
                                        !to return
  INTEGER(HSIZE_T), DIMENSION(ndims), INTENT(OUT) :: dims
                                        !Array containing sizes of
                                        !chunk dimensions
  INTEGER, INTENT(OUT) :: hdferr      !Error code
                                        !chunk rank on success and -1 on failure

END SUBROUTINE h5pget_chunk_f
```

---

**FORTRAN interface: h5pget\_class\_f**

```
SUBROUTINE h5pget_class_f(prp_id, classtype, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(OUT) :: classtype    !The type of the property list
                                        !to be created. Possible values
                                        !are:
                                        !H5P_NO_CLASS
                                        !H5P_FILE_CREATE_F
                                        !H5P_FILE_ACCESS_F
                                        !H5PE_DATASET_CREATE_F
                                        !H5P_DATASET_XFER_F
                                        !H5P_MOUNT_F
  INTEGER, INTENT(OUT) :: hdferr      !Error code
                                        !0 on success and -1 on failure

END SUBROUTINE h5pget_class_f
```

---

**FORTRAN interface: h5pget\_fill\_value\_f**

```
SUBROUTINE h5pget_fill_value_f(prp_id, type_id, fillvalue, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id    !Property list identifier
  INTEGER(HID_T), INTENT(IN) :: type_id   !Datatype identifier of
                                           !of fillvalue datatype
                                           !(in memory)
  TYPE(VOID), INTENT(IN) :: fillvalue    !Fillvalue
  INTEGER, INTENT(OUT) :: hdferr         !Error code
                                           !0 on success and -1 on failure

END SUBROUTINE h5pget_fill_value_f
```

**FORTRAN interface: h5pset\_chunk\_f**

```
SUBROUTINE h5pset_chunk_f(prp_id, ndims, dims, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(IN) :: ndims         !Number of chunk dimensions
  INTEGER(HSIZE_T), DIMENSION(ndims), INTENT(IN) :: dims
                                           !Array containing sizes of
                                           !chunk dimensions
  INTEGER, INTENT(OUT) :: hdferr       !Error code
                                           !0 on success and -1 on failure

END SUBROUTINE h5pset_chunk_f
```

**FORTRAN interface: h5pset\_deflate\_f**

```
SUBROUTINE h5pset_deflate_f(prp_id, level, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(IN) :: level        !Compression level
  INTEGER, INTENT(OUT) :: hdferr      !Error code
                                           ! 0 on success and -1 on failure

END SUBROUTINE h5pset_deflate_f
```

**FORTRAN interface: h5pset\_fill\_value\_f**

```
SUBROUTINE h5pset_fill_value_f(prp_id, type_id, fillvalue, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id    !Property list identifier
  INTEGER(HID_T), INTENT(IN) :: type_id   !Datatype identifier of
                                           !of fillvalue datatype
                                           !(in memory)
  TYPE(VOID), INTENT(IN) :: fillvalue    !Fillvalue
  INTEGER, INTENT(OUT) :: hdferr         !Error code
                                           !0 on success and -1 on failure

END SUBROUTINE h5pset_fill_value_f
```

---

**FORTRAN interface: h5pget\_version\_f**

```
SUBROUTINE h5pget_version_f(prp_id, boot, freelist, &
                             stab, shhdr, hdferr)

  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id      !Property list identifier
  INTEGER, DIMENSION(:), INTENT(OUT) :: boot !array to put boot
                                           !block version number
  INTEGER, DIMENSION(:), INTENT(OUT) :: freelist !array to put global
                                           !freelist version number

  INTEGER, DIMENSION(:), INTENT(OUT) :: stab !array to put symbol
                                           !table version number
  INTEGER, DIMENSION(:), INTENT(OUT) :: shhdr !array to put shared
                                           !object header version number
  INTEGER, INTENT(OUT) :: hdferr           !Error code

END SUBROUTINE h5pget_version_f
```

---

**FORTRAN interface: h5pset\_userblock\_f**

```
SUBROUTINE h5pset_userblock_f (prp_id, size, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER(HSIZE_T), INTENT(IN) :: size !Size of the user-block in bytes
  INTEGER, INTENT(OUT) :: hdferr      !Error code

END SUBROUTINE h5pset_userblock_f
```

---

**FORTRAN interface: h5pget\_userblock\_f**

```
SUBROUTINE h5pget_userblock_f(prp_id, block_size, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id      !Property list identifier
  INTEGER(HSIZE_T), DIMENSION(:), INTENT(OUT) :: block_size
                                          !Size of the
                                          !user-block in bytes
  INTEGER, INTENT(OUT) :: hdferr          !Error code

END SUBROUTINE h5pget_userblock_f
```

---

**FORTRAN interface: h5pset\_sizes\_f**

```
SUBROUTINE h5pset_sizes_f (prp_id, sizeof_addr, sizeof_size, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id      !Property list identifier
  INTEGER(SIZE_T), INTENT(IN) :: sizeof_addr !Size of an object
                                          !offset in bytes
  INTEGER(SIZE_T), INTENT(IN) :: sizeof_size !Size of an object
                                          !length in bytes
  INTEGER, INTENT(OUT) :: hdferr          !Error code

END SUBROUTINE h5pset_sizes_f
```

---

**FORTRAN interface: h5pget\_sizes\_f**

```
SUBROUTINE h5pget_sizes_f(prp_id, sizeof_addr, sizeof_size, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id      !Property list identifier
  INTEGER(SIZE_T), DIMENSION(:), INTENT(OUT) :: sizeof_addr
                                          !Size of an object
                                          !offset in bytes
  INTEGER(SIZE_T), DIMENSION(:), INTENT(OUT) :: sizeof_size
                                          !Size of an object
                                          !length in bytes
  INTEGER, INTENT(OUT) :: hdferr ! Error code

END SUBROUTINE h5pget_sizes_f
```

---

**FORTRAN interface: h5pset\_sym\_k\_f**

```
SUBROUTINE h5pset_sym_k_f (prp_id, ik, lk, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(IN) :: ik           !Symbol table tree rank
  INTEGER, INTENT(IN) :: lk           !Symbol table node size
  INTEGER, INTENT(OUT) :: hdferr      !Error code

END SUBROUTINE h5pset_sym_k_f
```

---

**FORTRAN interface: h5pget\_sym\_k\_f**

```
SUBROUTINE h5pget_sym_k_f(prp_id, ik, lk, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(OUT) :: ik           !Symbol table tree rank
  INTEGER, INTENT(OUT) :: lk           !Symbol table node size
  INTEGER, INTENT(OUT) :: hdferr      !Error code

END SUBROUTINE h5pget_sym_k_f
```

**FORTRAN interface: h5pset\_istore\_k\_f**

```
SUBROUTINE h5pset_istore_k_f (prp_id, ik, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(IN) :: ik           !1/2 rank of chunked storage B-tree
  INTEGER, INTENT(OUT) :: hdferr      !Error code

END SUBROUTINE h5pset_istore_k_f
```

---

**FORTRAN interface: h5pget\_istore\_k\_f**

```
SUBROUTINE h5pget_istore_k_f(prp_id, ik, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(OUT) :: ik           !1/2 rank of chunked storage B-tree
  INTEGER, INTENT(OUT) :: hdferr      !Error code

END SUBROUTINE h5pget_istore_k_f
```

---

**FORTRAN interface: h5pget\_driver\_f**

```
SUBROUTINE h5pget_driver_f(prp_id, driver, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(OUT) :: driver       !low-level file driver identifier
  INTEGER, INTENT(OUT) :: hdferr      !Error code

END SUBROUTINE h5pget_driver_f
```

---

**FORTRAN interface: h5pset\_alignment\_f**

```
SUBROUTINE h5pset_alignment_f(prp_id, threshold, alignment, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER(HSIZE_T), INTENT(IN) :: threshold !Threshold value
```

---

```
INTEGER(HSIZE_T), INTENT(IN) :: alignment !alignment value
INTEGER, INTENT(OUT) :: hdferr           !Error code

END SUBROUTINE h5pset_alignment_f
```

---

**FORTRAN interface: h5pget\_alignment\_f**

```
SUBROUTINE h5pget_alignment_f(prp_id, threshold, alignment, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id      !Property list identifier
  INTEGER(HSIZE_T), INTENT(OUT) :: threshold !Threshold value
  INTEGER(HSIZE_T), INTENT(OUT) :: alignment !alignment value
  INTEGER, INTENT(OUT) :: hdferr           !Error code

END SUBROUTINE h5pget_alignment_f
```

---

**FORTRAN interface: h5pset\_cache\_f**

```
SUBROUTINE h5pset_cache_f(prp_id, mdc_nelmts, rdcc_nelmts, rdcc_nbytes,
rdcc_w0, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(IN) :: mdc_nelmts    !Number of elements (objects)
                                       !in the meta data cache
  INTEGER, INTENT(IN) :: rdcc_nelmts   !Number of elements (objects)
                                       !in the meta data cache
  INTEGER(SIZE_T), INTENT(IN) :: rdcc_nbytes !Total size of the raw data
                                       !chunk cache, in bytes
  REAL, INTENT(IN) :: rdcc_w0          !Preemption policy
  INTEGER, INTENT(OUT) :: hdferr       !Error code

END SUBROUTINE h5pset_cache_f
```

---

**FORTRAN interface: h5pget\_cache\_f**

```
SUBROUTINE h5pget_cache_f(prp_id, mdc_nelmts, rdcc_nelmts, rdcc_nbytes,
rdcc_w0, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
```

```

INTEGER, INTENT(OUT) :: mdc_nelmts      !Number of elements (objects)
                                       !in the meta data cache
INTEGER, INTENT(OUT) :: rdcc_nelmts     !Number of elements (objects)
                                       !in the meta data cache
INTEGER(SIZE_T), INTENT(OUT) :: rdcc_nbytes !Total size of the raw data
                                       !chunk cache, in bytes
REAL, INTENT(OUT) :: rdcc_w0           !Preemption policy
INTEGER, INTENT(OUT) :: hdferr         !Error code

END SUBROUTINE h5pget_cache_f

```

---

### **FORTRAN interface: h5pset\_split\_f**

```

SUBROUTINE h5pset_split_f(prp_id, meta_ext, meta_plist, raw_ext,
raw_plist, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id      !Property list identifier
  CHARACTER(LEN=*), INTENT(IN) :: meta_ext  !Name of the extension for
                                       !the metafile filename
  INTEGER(HID_T), INTENT(IN) :: meta_plist  !Identifier of the meta file
                                       !access property list
  CHARACTER(LEN=*), INTENT(IN) :: raw_ext  !Name extension for the raw
file filename
  INTEGER(HID_T), INTENT(IN) :: raw_plist  !Identifier of the raw file
                                       !access property list
  INTEGER, INTENT(OUT) :: hdferr ! Error code

END SUBROUTINE h5pset_split_f

```

---

### **FORTRAN interface: h5pget\_split\_f**

```

SUBROUTINE h5pget_split_f(prp_id, meta_ext_size, meta_ext,
                           meta_plist, raw_ext_size, &
                           raw_ext, raw_plist, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER(SIZE_T), INTENT(IN) :: meta_ext_size
                                       !Number of characters of the meta
                                       !file extension to be copied to the
                                       !meta_ext buffer

```

```
CHARACTER(LEN=*), INTENT(OUT) :: meta_ext !Name of the extension for
                                         !the metafile filename
INTEGER(HID_T), INTENT(OUT) :: meta_plist !Identifier of the meta file
                                         !access property list
INTEGER(SIZE_T), INTENT(IN) :: raw_ext_size
                                         !Number of characters of the raw
                                         !file extension to be copied to the
                                         !raw_ext buffer
CHARACTER(LEN=*), INTENT(OUT) :: raw_ext
                                         !Name extension for the raw file filename
INTEGER(HID_T), INTENT(OUT) :: raw_plist
                                         !Identifier of the raw file
                                         !access property list
INTEGER, INTENT(OUT) :: hdferr !Error code

END SUBROUTINE h5pget_split_f
```

---

**FORTRAN interface: h5pset\_gc\_references\_f**

```
SUBROUTINE h5pset_gc_references_f (prp_id, gc_reference, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(IN) :: gc_reference !the flag for garbage collecting
                                         !references for the file
  INTEGER, INTENT(OUT) :: hdferr !Error code

END SUBROUTINE h5pset_gc_references_f
```

---

**FORTRAN interface: h5pget\_gc\_references\_f**

```
SUBROUTINE h5pget_gc_references_f (prp_id, gc_reference, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(OUT) :: gc_reference !the flag for garbage collecting
                                         !references for the file
  INTEGER, INTENT(OUT) :: hdferr !Error code

END SUBROUTINE h5pget_gc_references_f
```

---

**FORTRAN interface: h5pset\_layout\_f**

```

SUBROUTINE h5pset_layout_f (prp_id, layout, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(IN) :: layout      !Type of storage layout for raw data
                                     !possible values are:
                                     !H5D_COMPACT_F(0)
                                     !H5D_CONTIGUOUS_F(1)
                                     !H5D_CHUNKED_F(2)

  INTEGER, INTENT(OUT) :: hdferr      !Error code
END SUBROUTINE h5pset_layout_f

```

**FORTRAN interface: h5pget\_layout\_f**

```

SUBROUTINE h5pget_layout_f (prp_id, layout, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(OUT) :: layout      !Type of storage layout for raw data
                                     !possible values are:
                                     !H5D_COMPACT_F(0)
                                     !H5D_CONTIGUOUS_F(1)
                                     !H5D_CHUNKED_F(2)

  INTEGER, INTENT(OUT) :: hdferr      !Error code

```

**FORTRAN interface: h5pset\_filter\_f**

```

SUBROUTINE h5pset_filter_f(prp_id, filter, flags, cd_nelmts, cd_values,
                           hdferr)

  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(IN) :: filter      !Filter to be added to the pipeline.
  INTEGER, INTENT(IN) :: flags      !Bit vector specifying certain
                                     !general properties of the filter.

  INTEGER(SIZE_T), INTENT(IN) :: cd_nelmts
                                     !Number of elements in cd_values.

  INTEGER, DIMENSION(*), INTENT(IN) :: cd_values
                                     !Auxiliary data for the filter.

```

```
INTEGER, INTENT(OUT) :: hdferr      !Error code

END SUBROUTINE h5pset_filter_f
```

---

**FORTRAN interface: h5pget\_filter\_f**

```
SUBROUTINE h5pget_filter_f(prp_id, filter_number, flags, cd_nelmts,
                           cd_values, namelen, name, filter_id, hdferr)

IMPLICIT NONE
INTEGER(HID_T), INTENT(IN) :: prp_id    !Property list identifier
INTEGER, INTENT(IN) :: filter_number    !Sequence number within the filter
                                           !pipeline of the filter for which
                                           !information is sought
INTEGER, DIMENSION(*), INTENT(OUT) :: cd_values
                                           !Auxiliary data for the filter.
INTEGER, INTENT(OUT) :: flags           !Bit vector specifying certain general
                                           !properties of the filter.
INTEGER(SIZE_T), INTENT(INOUT) :: cd_nelmts
                                           !Number of elements in cd_values.
INTEGER(SIZE_T), INTENT(IN) :: namelen
                                           !Anticipated number of characters in
                                           !name.
CHARACTER(LEN=*), INTENT(OUT) :: name  !Name of the filter
INTEGER, INTENT(OUT) :: filter_id      !filter identification number
INTEGER, INTENT(OUT) :: hdferr         !Error code

END SUBROUTINE h5pget_filter_f
```

---

**FORTRAN interface: h5pset\_external\_f**

```
SUBROUTINE h5pset_external_f(prp_id, name, offset, bytes, hdferr)
IMPLICIT NONE
INTEGER(HID_T), INTENT(IN) :: prp_id    !Property list identifier
CHARACTER(LEN=*), INTENT(IN) :: name    !Name of an external file
INTEGER, INTENT(IN) :: offset           !Offset, in bytes, from the beginning
                                           !of the file to the location in the file
                                           !where the data starts.
INTEGER(HSIZE_T), INTENT(IN) :: bytes   !Number of bytes reserved in the
                                           !file for the data
```

---

```

        INTEGER, INTENT(OUT) :: hdferr      !Error code

    END SUBROUTINE h5pset_external_f

```

---

**FORTRAN interface: h5pget\_external\_count\_f**

```

SUBROUTINE h5pget_external_count_f (prp_id, count, hdferr)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: prp_id    !Property list identifier
    INTEGER, INTENT(OUT) :: count          !number of external files for the
                                           !specified dataset
    INTEGER, INTENT(OUT) :: hdferr        !Error code

    END SUBROUTINE h5pget_external_count_f

```

---

**FORTRAN interface: h5pget\_external\_f**

```

SUBROUTINE h5pget_external_f(prp_id, idx, name_size, name, offset,bytes,
                             hdferr)

    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: prp_id    !Property list identifier
    INTEGER, INTENT(IN) :: idx              !External file index.
    INTEGER, INTENT(IN) :: name_size        !Maximum length of name array
    CHARACTER(LEN=*), INTENT(OUT) :: name   !Name of an external file
    INTEGER, INTENT(OUT) :: offset          !Offset, in bytes, from the beginning
                                           !of the file to the location in the file
                                           !where the data starts.
    INTEGER(HSIZE_T), INTENT(OUT) :: bytes  !Number of bytes reserved in the
                                           !file for the data
    INTEGER, INTENT(OUT) :: hdferr        !Error code

    END SUBROUTINE h5pget_external_f

```

---

**FORTRAN interface: h5pset\_hyper\_cache\_f**

```

SUBROUTINE h5pset_hyper_cache_f(prp_id, cache, limit, hdferr)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: prp_id    !Property list identifier
    INTEGER, INTENT(IN) :: cache            !

```

---

```
INTEGER, INTENT(IN) :: limit      !Maximum size of the hyperslab block to
                                   !cache. 0 (zero) indicates no limit.
INTEGER, INTENT(OUT) :: hdferr   !Error code

END SUBROUTINE h5pset_hyper_cache_f
```

---

**FORTRAN interface: h5pget\_hyper\_cache\_f**

```
SUBROUTINE h5pget_hyper_cache_f(prp_id, cache, limit, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(OUT) :: cache      !
  INTEGER, INTENT(OUT) :: limit      !Maximum size of the hyperslab block to
                                   !cache. 0 (zero) indicates no limit.
  INTEGER, INTENT(OUT) :: hdferr    !Error code

END SUBROUTINE h5pget_hyper_cache_f
```

---

**FORTRAN interface: h5pset\_btree\_ratios\_f**

```
SUBROUTINE h5pset_btree_ratios_f(prp_id, left, middle, right, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  REAL, INTENT(IN) :: left    !The B-tree split ratio for left-most nodes.
  REAL, INTENT(IN) :: middle !The B-tree split ratio for all other nodes
  REAL, INTENT(IN) :: right   !The B-tree split ratio for right-most
                              !nodes and lone nodes.
  INTEGER, INTENT(OUT) :: hdferr !Error code

END SUBROUTINE h5pset_btree_ratios_f
```

---

**FORTRAN interface: h5pget\_btree\_ratios\_f**

```
SUBROUTINE h5pget_btree_ratios_f(prp_id, left, middle, right, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  REAL, INTENT(OUT) :: left    !The B-tree split ratio for left-most nodes.
  REAL, INTENT(OUT) :: middle !The B-tree split ratio for all other nodes
  REAL, INTENT(OUT) :: right   !The B-tree split ratio for right-most
```

---

!nodes and lone nodes.

```

INTEGER, INTENT(OUT) :: hdferr !Error code

END SUBROUTINE h5pget_btree_ratios_f

```

---

#### **FORTRAN interface: h5pset\_fapl\_mpi\_f**

```

SUBROUTINE h5pset_fapl_mpi_f(prp_id, comm, info, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(IN) :: comm      MPI communicator to be used for file open
                                   !as defined in MPI_FILE_OPEN of MPI-2
  INTEGER, INTENT(IN) :: info      !MPI info object to be used for file open
                                   !as defined in MPI_FILE_OPEN of MPI-2
  INTEGER, INTENT(OUT) :: hdferr !Error code
END SUBROUTINE h5pset_fapl_mpi_f

```

---

#### **FORTRAN interface: h5pget\_fapl\_mpi\_f**

```

SUBROUTINE h5pget_fapl_mpi_f(prp_id, comm, info, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(OUT) :: comm          !buffer to return communicator
  INTEGER, INTENT(IN) :: info          !buffer to return info object
                                   !as defined in MPI_FILE_OPEN of MPI-2
  INTEGER, INTENT(OUT) :: hdferr       !Error code
END SUBROUTINE h5pget_fapl_mpi_f

```

---

#### **FORTRAN interface: h5pset\_dxpl\_mpi\_f**

```

SUBROUTINE h5pset_dxpl_mpi_f(prp_id, data_xfer_mode, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(IN) :: data_xfer_mode
                                   !Data transfer mode. Possible values are:
                                   !H5FD_MPIO_INDEPENDENT_F (0)
                                   !H5FD_MPIO_COLLECTIVE_F (1)
  INTEGER, INTENT(OUT) :: hdferr !Error code

```

```
END SUBROUTINE h5pset_dxpl_mpi_f
```

---

**FORTRAN interface: h5pget\_dxpl\_mpi\_f**

```
SUBROUTINE h5pget_dxpl_mpi_f(prp_id, data_xfer_mode, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: prp_id !Property list identifier
  INTEGER, INTENT(OUT) :: data_xfer_mode
                                !Data transfer mode. Possible values are:
                                !H5FD_MPIO_INDEPENDENT_F (0)
                                !H5FD_MPIO_COLLECTIVE_F (1)
  INTEGER, INTENT(OUT) :: hdferr      !Error code
END SUBROUTINE h5pget_dxpl_mpi_f
```

---

# The FORTRAN 90 API to HDF5

## h5r: References

### FORTRAN interface: h5rcreate\_f

#### To create an object reference

```

SUBROUTINE h5rcreate_f(loc_id, name, ref, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: loc_id    !Location identifier
  CHARACTER(LEN=*), INTENT(IN) :: name
                                     !Name of the object at location specified
                                     !by loc_id identifier
  TYPE(hobj_ref_t_f), INTENT(OUT) :: ref !Object reference
  INTEGER, INTENT(OUT) :: hdferr        !Error code

END SUBROUTINE h5rcreate_f

```

#### To create a region reference

```

SUBROUTINE h5rcreate_f(loc_id, name, space_id, ref, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: loc_id    !Location identifier
  CHARACTER(LEN=*), INTENT(IN) :: name
                                     !Name of the dataset at location specified
                                     !by loc_id identifier
  INTEGER(HID_T), INTENT(IN) :: space_id !Dataset's dataspace identifier
  TYPE(hdset_reg_ref_t_f), INTENT(OUT) :: ref !Dataset region reference
  INTEGER, INTENT(OUT) :: hdferr        !Error code

END SUBROUTINE h5rcreate_f

```

### FORTRAN interface: h5rdereference\_f

#### To dereference an object

```

SUBROUTINE h5rdereference_f(dset_id, ref, obj_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: dset_id  !Dataset identifier
  TYPE(hobj_ref_t_f), INTENT(IN) :: ref  !Object reference
  INTEGER(HID_T), INTENT(OUT) :: obj_id  !Object identifier
  INTEGER, INTENT(OUT) :: hdferr        !Error code

```

```
END SUBROUTINE h5rdereference_f
```

**To dereference a region**

```
SUBROUTINE h5rdereference_f(dset_id, ref, obj_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: dset_id      !Dataset identifier
  TYPE(hdset_reg_ref_t_f), INTENT(IN) :: ref !Object reference
  INTEGER(HID_T), INTENT(OUT) :: obj_id     !Object identifier
  INTEGER, INTENT(OUT) :: hdferr           !Error code

END SUBROUTINE h5rdereference_f
```

---

**FORTTRAN interface: h5rget\_region\_f**

```
SUBROUTINE h5rget_region_f(dset_id, ref, space_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: dset_id      !Dataset identifier
  TYPE(hdset_reg_ref_t_f), INTENT(IN) :: ref !Dataset region reference
  INTEGER(HID_T), INTENT(OUT) :: space_id   !Space identifier
  INTEGER, INTENT(OUT) :: hdferr           !Error code

END SUBROUTINE h5rget_region_f
```

---

**FORTTRAN interface: h5rget\_object\_type\_f**

```
SUBROUTINE h5rget_object_type_f(dset_id, ref, obj_type, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: dset_id      !Dataset identifier
  TYPE(hobj_ref_t_f), INTENT(IN) :: ref     !Object reference
  INTEGER, INTENT(OUT) :: obj_type          !Object type
                                          !H5G_UNKNOWN_F      (-1)
                                          !H5G_LINK_F        0
                                          !H5G_GROUP_F       1
                                          !H5G_DATASET_F    2
                                          !H5G_TYPE_F       3

  INTEGER, INTENT(OUT) :: hdferr           !Error code

END SUBROUTINE h5rget_object_type_f
```

---

## The FORTRAN 90 API to HDF5

### h5s: Dataspaces

---

**FORTRAN interface: h5sclose\_f**

```

SUBROUTINE h5sclose_f(space_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: space_id  !Dataspace identifier
  INTEGER, INTENT(OUT) :: hdferr          !Error code
                                          !0 on success and -1 on failure

END SUBROUTINE h5sclose_f

```

---

**FORTRAN interface: h5scopy\_f**

```

SUBROUTINE h5scopy_f(space_id, new_space_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: space_id  !Dataspace identifier
  INTEGER(HID_T), INTENT(OUT) :: new_space_id
                                          !Identifier of dataspace's copy
  INTEGER, INTENT(OUT) :: hdferr          !Error code
                                          !0 on success and -1 on failure

END SUBROUTINE h5scopy_f

```

---

**FORTRAN interface: h5screate\_f**

```

SUBROUTINE h5screate_f(classtype, space_id, hdferr)
  IMPLICIT NONE
  INTEGER, INTENT(IN) :: classtype        !The type of the dataspace
                                          !to be created. Possible values
                                          !are:
                                          !H5S_SCALAR_F
                                          !H5S_SIMPLE_F
  INTEGER(HID_T), INTENT(OUT) :: space_id !Dataspace identifier
  INTEGER, INTENT(OUT) :: hdferr          !Error code
                                          !0 on success and -1 on failure

```

---

```
END SUBROUTINE h5screate_f
```

---

**FORTRAN interface: h5screate\_simple\_f**

```
SUBROUTINE h5screate_simple_f(rank, dims, space_id, hdferr, maxdims)
  IMPLICIT NONE
  INTEGER, INTENT(IN) :: rank          !Number of dataspace dimensions
  INTEGER(HSIZE_T), INTENT(IN) :: dims(*) !Array with the dimension
                                          !sizes
  INTEGER(HID_T), INTENT(OUT) :: space_id !Dataspace identifier
  INTEGER, INTENT(OUT) :: hdferr        !Error code
                                          !0 on success and -1 on failure
  INTEGER(HSIZE_T), OPTIONAL, INTENT(IN) :: maxdims(*)
                                          !Array with the maximum
                                          !dimension sizes

END SUBROUTINE h5screate_simple_f
```

---

**FORTRAN interface: h5sextent\_copy\_f**

```
SUBROUTINE h5sextent_copy_f(dest_space_id, source_space_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: dest_space_id !Identifier of destination
                                          !dataspace
  INTEGER(HID_T), INTENT(IN) :: source_space_id !Identifier of source
                                          !dataspace
  INTEGER, INTENT(OUT) :: hdferr             !Error code
                                          !0 on success and -1 on failure

END SUBROUTINE h5sextent_copy_f
```

---

**FORTRAN interface: h5sget\_select\_npoints\_f**

```
SUBROUTINE h5sget_select_npoints_f(space_id, npoints, hdferr)

  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: space_id !Dataspace identifier
  INTEGER(HSSIZE_T), INTENT(OUT) :: npoints !Number of elements in the
```

```

!selection
INTEGER, INTENT(OUT) :: hdferr      !Error code
!0 on success and -1 on failure

END SUBROUTINE h5sget_select_npoints_f

```

---

**FORTRAN interface: h5sget\_simple\_extent\_dims\_f**

```

SUBROUTINE h5sget_simple_extent_dims_f(space_id, dims, maxdims, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: space_id !Dataspace identifier
  INTEGER(HSIZE_T), DIMENSION(*), INTENT(OUT) :: dims
!Array to store dimension sizes
  INTEGER(HSIZE_T), DIMENSION(*), INTENT(OUT) :: maxdims
!Array to store max dimension
!sizes
  INTEGER, INTENT(OUT) :: hdferr      !Error code
!0 on success and -1 on failure

END SUBROUTINE h5sget_simple_extent_dims_f

```

---

**FORTRAN interface: h5sget\_simple\_extent\_ndims\_f**

```

SUBROUTINE h5sget_simple_extent_ndims_f(space_id, rank, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: space_id !Dataspace identifier
  INTEGER, INTENT(OUT) :: rank          !Number of dimensions
  INTEGER, INTENT(OUT) :: hdferr      !Error code
!0 on success and -1 on failure

END SUBROUTINE h5sget_simple_extent_ndims_f

```

---

**FORTRAN interface: h5sget\_simple\_extent\_npoints\_f**

```

SUBROUTINE h5sget_simple_extent_npoints_f(space_id, npoints, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: space_id !Dataspace identifier
  INTEGER(HSIZE_T), INTENT(OUT) :: npoints !Number of elements in
!dataspace

```

---

```
        INTEGER, INTENT(OUT) :: hdferr           !Error code
                                                !0 on success and -1 on failure

    END SUBROUTINE h5sget_simple_extent_npoints_f
```

---

**FORTRAN interface: h5sget\_simple\_extent\_type\_f**

```
    SUBROUTINE h5sget_simple_extent_type_f(space_id, classtype, hdferr)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: space_id !Dataspace identifier
    INTEGER, INTENT(OUT) :: classtype      !Class type , possible values
                                          !are:
                                          !H5S_NO_CLASS_F
                                          !H5S_SCALAR_F
                                          !H5S_SIMPLE_F

    INTEGER, INTENT(OUT) :: hdferr        !Error code
                                          !0 on success and -1 on failure

    END SUBROUTINE h5sget_simple_extent_type_f
```

---

**FORTRAN interface: h5sis\_simple\_f**

```
    SUBROUTINE h5sis_simple_f(space_id, flag, hdferr)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: space_id !Dataspace identifier
    LOGICAL, INTENT(OUT) :: flag          !Flag, idicates if dataspace
                                          !is simple or not ( TRUE or
                                          !FALSE)

    INTEGER, INTENT(OUT) :: hdferr        !Error code
                                          !0 on success and -1 on failure

    END SUBROUTINE h5sis_simple_f
```

---

**FORTRAN interface: h5soffset\_simple\_f**

```
    SUBROUTINE h5soffset_simple_f(space_id, offset, hdferr)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: space_id !Dataspace identifier
    INTEGER(HSSIZE_T), DIMENSION(*), INTENT(IN) :: offset
```

---

```

!The offset at which to position
!the selection
INTEGER, INTENT(OUT) :: hdferr      !Error code
!0 on success and -1 on failure

END SUBROUTINE h5soffset_simple_f

```

---

**FORTRAN interface: h5sselect\_all\_f**

```

SUBROUTINE h5sselect_all_f(space_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: space_id !Dataspace identifier
  INTEGER, INTENT(OUT) :: hdferr        !Error code
!0 on success and -1 on failure

END SUBROUTINE h5sselect_all_f

```

---

**FORTRAN interface: h5sselect\_elements\_f**

```

SUBROUTINE h5sselect_elements_f(space_id, operator, num_elements, &
                                coord, hdferr)

  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: space_id !Dataspace identifier
  INTEGER, INTENT(IN) :: op             !Flag, valid values are:
!H5S_SELECT_SET_F
!H5S_SELECT_OR_F
  INTEGER, INTENT(IN) :: num_elements  !Number of elements to be
!selected
  INTEGER(HSSIZE_T), DIMENSION(*,*), INTENT(IN) :: coord
!Array with the coordinates
!of the selected elements
!coord(num_elements, rank)
  INTEGER, INTENT(OUT) :: hdferr      !Error code
!0 on success and -1 on failure

END SUBROUTINE h5sselect_elements_f

```

**FORTRAN interface: h5sselect\_hyperslab\_f**

```
SUBROUTINE h5sselect_hyperslab_f(space_id, operator, start, count, &
                                hdferr, stride, block)

    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: space_id !Dataspace identifier
    INTEGER, INTENT(IN) :: op              !Flag, valid values are:
                                          !H5S_SELECT_SET_F
                                          !H5S_SELECT_OR_F
                                          !
    INTEGER(HSSIZE_T), DIMENSION(*), INTENT(IN) :: start
                                          !Starting coordinates of the hyperslab
    INTEGER(HSIZE_T), DIMENSION(*), INTENT(IN) :: count
                                          !Number of blocks to select
                                          !from dataspace
    INTEGER, INTENT(OUT) :: hdferr        !Error code
                                          !0 on success and -1 on failure
    INTEGER(HSIZE_T), DIMENSION(*), OPTIONAL, INTENT(IN) :: stride
                                          !Array of how many elements to move
                                          !in each direction
    INTEGER(HSIZE_T), DIMENSION(*), OPTIONAL, INTENT(IN) :: block
                                          !Size of the element block

END SUBROUTINE h5sselect_hyperslab_f
```

---

**FORTRAN interface: h5sselect\_none\_f**

```
SUBROUTINE h5sselect_none_f(space_id, hdferr)

    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: space_id !Dataspace identifier
    INTEGER, INTENT(OUT) :: hdferr        !Error code
                                          !0 on success and -1 on failure

END SUBROUTINE h5sselect_none_f
```

**FORTRAN interface: h5sselect\_valid\_f**

```

SUBROUTINE h5sselect_valid_f(space_id, flag, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: space_id !Dataspace identifier
  LOGICAL, INTENT(OUT) :: flag           !TRUE if the selection is
                                         !contained within the extent,
                                         !FALSE otherwise.
  INTEGER, INTENT(OUT) :: hdferr         !Error code
                                         !0 on success and -1 on failure
END SUBROUTINE h5sselect_valid_f

```

**FORTRAN interface: h5sset\_extent\_none\_f**

```

SUBROUTINE h5sset_extent_none_f(space_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: space_id !Dataspace identifier
  INTEGER, INTENT(OUT) :: hdferr        !Error code
                                         !0 on success and -1 on failure
END SUBROUTINE h5sset_extent_none_f

```

**FORTRAN interface: h5sset\_extent\_simple\_f**

```

SUBROUTINE h5sset_extent_simple_f(space_id, rank, current_size, &
                                  maximum_size, hdferr)

  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: space_id !Dataspace identifier
  INTEGER, INTENT(IN) :: rank           !Dataspace rank
  INTEGER(HSIZE_T), DIMENSION(rank), INTENT(IN) :: current_size
                                         !Array with the new sizes
                                         !of dimensions
  INTEGER(HSIZE_T), DIMENSION(rank), INTENT(IN) ::
                                         !Array with the new maximum
                                         !sizes of dimensions
  INTEGER, INTENT(OUT) :: hdferr        !Error code
                                         !0 on success and -1 on failure

```

```
END SUBROUTINE h5sset_extent_simple_f
```

---

# The FORTRAN 90 API to HDF5

## h5t: Datatypes

---

### FORTRAN interface: h5tclose\_f

```

SUBROUTINE h5tclose_f(type_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(OUT) :: hdferr      !Error code
                                      !0 on success and -1 on failure
END SUBROUTINE h5tclose_f

```

---

### FORTRAN interface: h5tcommit\_f

```

SUBROUTINE h5tcommit_f(loc_id, name, type_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: loc_id !File or group identifier
  CHARACTER(LEN=*), INTENT(IN) :: name !Datatype name within file or
                                      !group
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(OUT) :: hdferr      !Error code
                                      !0 on success and -1 on failure
END SUBROUTINE h5tcommit_f

```

---

### FORTRAN interface: h5tcopy\_f

```

SUBROUTINE h5tcopy_f(type_id, new_type_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id      !Datatype identifier
  INTEGER(HID_T), INTENT(OUT) :: new_type_id !Identifier of datatype's
                                      !copy
  INTEGER, INTENT(OUT) :: hdferr          !Error code
                                      !0 on success and -1 on failure
END SUBROUTINE h5tcopy_f

```

---

### FORTRAN interface: h5tget\_class\_f

```
SUBROUTINE h5tget_class_f(type_id, class, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(OUT) :: class      !Datatype class, possible values are:
                                     !H5T_NO_CLASS_F
                                     !H5T_INTEGER_F
                                     !H5T_FLOAT_F
                                     !H5T_TIME_F
                                     !H5T_STRING_F
                                     !H5T_BITFIELD_F
                                     !H5T_OPAQUE_F
                                     !H5T_COMPOUND_F
                                     !H5T_REFERENCE_F
                                     !H5T_ENUM_F

  INTEGER, INTENT(OUT) :: hdferr      !Error code
                                     !0 on success and -1 on failure

END SUBROUTINE h5tget_class_f
```

---

**FORTRAN interface: h5tget\_order\_f**

```
SUBROUTINE h5tget_order_f(type_id, order, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(OUT) :: order !Datatype byte order, possible values
                                !are:
                                !H5T_ORDER_LE_F
                                !H5T_ORDER_BE_F
                                !H5T_ORDER_VAX_F

  INTEGER, INTENT(OUT) :: hdferr      !Error code
                                !0 on success and -1 on failure

END SUBROUTINE h5tget_order_f
```

---

**FORTRAN interface: h5tget\_size\_f**

```
SUBROUTINE h5tget_size_f(type_id, size, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(OUT) :: size      !Datatype size
  INTEGER, INTENT(OUT) :: hdferr      !Error code
                                !0 on success and -1 on failure
```

---

```
END SUBROUTINE h5tget_size_f
```

---

**FORTRAN interface: h5topen\_f**

```

SUBROUTINE h5topen_f(loc_id, name, type_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: loc_id      !File or group identifier
  CHARACTER(LEN=*), INTENT(IN) :: name     !Datatype name within file or
                                           !group
  INTEGER(HID_T), INTENT(out) :: type_id  !Datatype identifier
  INTEGER, INTENT(OUT) :: hdferr          !Error code
                                           !0 on success and -1 on failure
END SUBROUTINE h5topen_f

```

---

**FORTRAN interface: h5tset\_order\_f**

```

SUBROUTINE h5tset_order_f(type_id, order, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(IN) :: order        !Datatype byte order, possible values
                                     !are:
                                     !H5T_ORDER_LE_F
                                     !H5T_ORDER_BE_F
                                     !H5T_ORDER_VAX_F
  INTEGER, INTENT(OUT) :: hdferr      !Error code
                                     !0 on success and -1 on failure
END SUBROUTINE h5tset_order_f

```

---

**FORTRAN interface: h5tset\_size\_f**

```

SUBROUTINE h5tset_size_f(type_id, size, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(IN) :: size          !Datatype size
  INTEGER, INTENT(OUT) :: hdferr       !Error code
                                     !0 on success and -1 on failure
END SUBROUTINE h5tset_size_f

```

---

**FORTRAN interface: h5tequal\_f**

```
SUBROUTINE h5tequal_f(type1_id, type2_id, flag, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type1_id !Datatype identifier
  INTEGER(HID_T), INTENT(IN) :: type2_id !Datatype identifier
  LOGICAL, INTENT(OUT) :: flag          !TRUE/FALSE flag to indicate if two
                                         !datatypes are equal
  INTEGER, INTENT(OUT) :: hdferr        !Error code

END SUBROUTINE h5tequal_f
```

---

**FORTRAN interface: h5tcommitted\_f**

```
SUBROUTINE h5tcommitted_f(type_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(OUT) :: hdferr        !Error code

END SUBROUTINE h5tcommitted_f
```

---

**FORTRAN interface: h5tget\_precision\_f**

```
SUBROUTINE h5tget_precision_f(type_id, precision, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(OUT) :: precision     !Datatype precision
  INTEGER, INTENT(OUT) :: hdferr        !Error code

END SUBROUTINE h5tget_precision_f
```

---

**FORTTRAN interface: h5tset\_precision\_f**

```
SUBROUTINE h5tset_precision_f(type_id, precision, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(IN) :: precision      !Datatype precision
  INTEGER, INTENT(OUT) :: hdferr        !Error code

END SUBROUTINE h5tset_precision_f
```

---

**FORTTRAN interface: h5tget\_offset\_f**

```
SUBROUTINE h5tget_offset_f(type_id, offset, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(OUT) :: offset        !Datatype bit offset of the
                                        !first significant bit
  INTEGER, INTENT(OUT) :: hdferr        !Error code

END SUBROUTINE h5tget_offset_f
```

---

**FORTTRAN interface: h5tset\_offset\_f**

```
SUBROUTINE h5tset_offset_f(type_id, offset, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(IN) :: offset        !Datatype bit offset of the
                                        !first significant bit
  INTEGER, INTENT(OUT) :: hdferr        !Error code

END SUBROUTINE h5tset_offset_f
```

---

**FORTTRAN interface: h5tget\_pad\_f**

```
SUBROUTINE h5tget_pad_f(type_id, lsbpad, msbpad, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
```

---

```
INTEGER, INTENT(OUT) :: lsbpad      !padding type of the
                                     !least significant bit
INTEGER, INTENT(OUT) :: msbpad      !padding type of the
                                     !most significant bit
                                     !Possible values of padding type are:
                                     !H5T_PAD_ZERO_F = 0
                                     !H5T_PAD_ONE_F = 1
                                     !H5T_PAD_BACKGROUND_F = 2
                                     !H5T_PAD_ERROR_F = -1
                                     !H5T_PAD_NPAD_F = 3

INTEGER, INTENT(OUT) :: hdferr      !Error code

END SUBROUTINE h5tget_pad_f
```

---

**FORTRAN interface: h5tset\_pad\_f**

```
SUBROUTINE h5tset_pad_f(type_id, lsbpad, msbpad, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(IN) :: lsbpad        !padding type of the
                                       !least significant bit
  INTEGER, INTENT(IN) :: msbpad        !padding type of the
                                       !most significant bit
                                       !Possible values of padding type are:
                                       !H5T_PAD_ZERO_F = 0
                                       !H5T_PAD_ONE_F = 1
                                       !H5T_PAD_BACKGROUND_F = 2
                                       !H5T_PAD_ERROR_F = -1
                                       !H5T_PAD_NPAD_F = 3

  INTEGER, INTENT(OUT) :: hdferr      !Error code

END SUBROUTINE h5tset_pad_f
```

---

**FORTRAN interface: h5tget\_sign\_f**

```
SUBROUTINE h5tget_sign_f(type_id, sign, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(OUT) :: sign    !sign type for an integer type
```

---

```

!possible values are:
!Unsigned integer type H5T_SGN_NONE_F = 0
!Two's complement signed integer type
!H5T_SGN_2_F = 1
!or error value: H5T_SGN_ERROR_F=-1
    INTEGER, INTENT(OUT) :: hdferr          !Error code

END SUBROUTINE h5tget_sign_f

```

---

### **FORTRAN interface: h5tset\_sign\_f**

```

SUBROUTINE h5tset_sign_f(type_id, sign, hdferr)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
    INTEGER, INTENT(IN) :: sign           !sign type for an integer type
                                           !possible values are:
                                           !Unsigned integer type H5T_SGN_NONE_F = 0
                                           !Two's complement signed integer type
                                           !H5T_SGN_2_F = 1
                                           !or error value: H5T_SGN_ERROR_F=-1
    INTEGER, INTENT(OUT) :: hdferr       !Error code

END SUBROUTINE h5tset_sign_f

```

---

### **FORTRAN interface: h5tget\_fields\_f**

```

SUBROUTINE h5tget_fields_f(type_id, epos, esize, mpos, msize, hdferr)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
    INTEGER, INTENT(OUT) :: epos          !exponent bit-position
    INTEGER, INTENT(OUT) :: esize        !size of exponent in bits
    INTEGER, INTENT(OUT) :: mpos         !mantissa bit-position
    INTEGER, INTENT(OUT) :: msize        !size of mantissa in bits
    INTEGER, INTENT(OUT) :: hdferr       !Error code

END SUBROUTINE h5tget_fields_f

```

**FORTRAN interface: h5tset\_fields\_f**

```
SUBROUTINE h5tset_fields_f(type_id, epos, esize, mpos, msize, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(IN) :: epos          !exponent bit-position
  INTEGER, INTENT(IN) :: esize        !size of exponent in bits
  INTEGER, INTENT(IN) :: mpos        !mantissa bit-position
  INTEGER, INTENT(IN) :: msize        !size of mantissa in bits
  INTEGER, INTENT(OUT) :: hdferr      !Error code

END SUBROUTINE h5tset_fields_f
```

---

**FORTRAN interface: h5tget\_ebias\_f**

```
SUBROUTINE h5tget_ebias_f(type_id, ebias, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(OUT) :: ebias         !Datatype exponent bias of a
                                       !floating- point type
  INTEGER, INTENT(OUT) :: hdferr       !Error code

END SUBROUTINE h5tget_ebias_f
```

---

**FORTRAN interface: h5tset\_ebias\_f**

```
SUBROUTINE h5tset_ebias_f(type_id, ebias, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(IN) :: ebias         !Datatype exponent bias of a
                                       !floating-point type, which can't be
  INTEGER, INTENT(OUT) :: hdferr       !Error code

END SUBROUTINE h5tset_ebias_f
```

---

**FORTRAN interface: h5tget\_norm\_f**

```
SUBROUTINE h5tget_norm_f(type_id, norm, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(OUT) :: norm          !mantissa normalization of a
                                       !floating-point datatype
                                       !Valid normalization types are:
                                       !H5T_NORM_IMPLIED_F(0),MSB of mantissa is
                                       !not
                                       !stored, always 1, H5T_NORM_MSBSET_F(1),
                                       !MSB of
                                       !mantissa is always 1, H5T_NORM_NONE_F(2)
                                       !Mantissa is not normalize
  INTEGER, INTENT(OUT) :: hdferr       !Error code

END SUBROUTINE h5tget_norm_f
```

**FORTRAN interface: h5tset\_norm\_f**

```
SUBROUTINE h5tset_norm_f(type_id, norm, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(IN) :: norm !mantissa normalization of a floating-point
                              !datatype
                              !Valid normalization types are:
                              !H5T_NORM_IMPLIED_F(0),MSB of mantissa is
                              !not
                              !stored, always 1, H5T_NORM_MSBSET_F(1), MSB of
                              !mantissa is always 1, H5T_NORM_NONE_F(2)
                              !Mantissa is not normalize
  INTEGER, INTENT(OUT) :: hdferr       !Error code

END SUBROUTINE h5tset_norm_f
```

**FORTRAN interface: h5tget\_inpad\_f**

```
SUBROUTINE h5tget_inpad_f(type_id, padtype, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(OUT) :: padtype      !padding type for unused bits
                                       !in floating-point datatypes.
                                       !Possible values of padding type are:
                                       !H5T_PAD_ZERO_F = 0
                                       !H5T_PAD_ONE_F = 1
                                       !H5T_PAD_BACKGROUND_F = 2

  INTEGER, INTENT(OUT) :: hdferr      !Error code

END SUBROUTINE h5tget_inpad_f
```

---

**FORTRAN interface: h5tset\_inpad\_f**

```
IMPLICIT NONE
INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
INTEGER, INTENT(IN) :: padtype !padding type for unused bits
                                       !in floating-point datatypes.
                                       !Possible values of padding type are:
                                       !H5T_PAD_ZERO_F = 0
                                       !H5T_PAD_ONE_F = 1
                                       !H5T_PAD_BACKGROUND_F = 2

INTEGER, INTENT(OUT) :: hdferr !Error code

END SUBROUTINE h5tset_inpad_f
```

---

**FORTRAN interface: h5tget\_cset\_f**

```
SUBROUTINE h5tget_cset_f(type_id, cset, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(OUT) :: cset !character set type of a string datatype
                                       !Possible values of padding type are:
```

---

```

                                !H5T_CSET_ASCII_F = 0
    INTEGER, INTENT(OUT) :: hdferr  !Error code

    END SUBROUTINE h5tget_cset_f

```

---

**FORTRAN interface: h5tset\_cset\_f**

```

SUBROUTINE h5tset_cset_f(type_id, cset, hdferr)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
    INTEGER, INTENT(IN) :: cset  !character set type of a string datatype
                                !Possible values of padding type are:
                                !H5T_CSET_ASCII_F = 0
    INTEGER, INTENT(OUT) :: hdferr  !Error code

    END SUBROUTINE h5tset_cset_f

```

---

**FORTRAN interface: h5tget\_strpad\_f**

```

SUBROUTINE h5tget_strpad_f(type_id, strpad, hdferr)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
    INTEGER, INTENT(OUT) :: strpad  !string padding method for a
                                    !string datatype
                                    ! Possible values of padding type are:
                                    !Pad with zeros (as C does):
H5T_STR_NULL_F(0),
                                    !Pad with spaces (as FORTRAN does):
                                    !H5T_STR_SPACE_F(1)
    INTEGER, INTENT(OUT) :: hdferr  !Error code

    END SUBROUTINE h5tget_strpad_f

```

---

**FORTRAN interface: h5tset\_strpad\_f**

```
SUBROUTINE h5tset_strpad_f(type_id, strpad, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(IN) :: strpad !string padding method for a string
                                !datatype
                                !Possible values of padding type are:
                                !Pad with zeros (as C does):
                                !H5T_STR_NULL_F(0),
                                !Pad with spaces (as FORTRAN does):
                                !H5T_STR_SPACE_F(1)
  INTEGER, INTENT(OUT) :: hdferr      !Error code

END SUBROUTINE h5tset_strpad_f
```

---

**FORTRAN interface: h5tget\_nmembers\_f**

```
SUBROUTINE h5tget_nmembers_f(type_id, num_members, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(OUT) :: num_members !number of fields in a compound
                                !datatype
  INTEGER, INTENT(OUT) :: hdferr      !Error code

END SUBROUTINE h5tget_nmembers_f
```

---

**FORTRAN interface: h5tget\_member\_name\_f**

```
SUBROUTINE h5tget_member_name_f(type_id, index, member_name, namelen,
                                hdferr)

  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(IN) :: index !Field index (0-based) of the field name to
                                !retrieve
  CHARACTER(LEN=*), INTENT(OUT) :: member_name !name of a field of
                                                !a compound datatype
  INTEGER, INTENT(OUT) :: namelen      !Length the name
```

---

```

        INTEGER, INTENT(OUT) :: hdferr          !Error code

    END SUBROUTINE h5tget_member_name_f

```

---

**FORTRAN interface: h5tget\_member\_offset\_f**

```

SUBROUTINE h5tget_member_offset_f(type_id, member_no, offset, hdferr)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: type_id  !Datatype identifier
    INTEGER, INTENT(IN) :: member_no      !Number of the field
                                          !whose offset is requested
    INTEGER(SIZE_T), INTENT(OUT) :: offset !byte offset of the the beginning
                                          !of the field
    INTEGER, INTENT(OUT) :: hdferr        !Error code

END SUBROUTINE h5tget_member_offset_f

```

---

**FORTRAN interface: h5tget\_member\_type\_f**

```

SUBROUTINE h5tget_member_type_f(type_id, field_idx, datatype, hdferr)
    IMPLICIT NONE
    INTEGER(HID_T), INTENT(IN) :: type_id  !Datatype identifier
    INTEGER, INTENT(IN) :: field_idx !Field index (0-based) of the field type
                                          !to retrieve
    INTEGER(HID_T), INTENT(OUT) :: datatype !identifier of a copy of
                                          !the datatype of the field
    INTEGER, INTENT(OUT) :: hdferr        !Error code

END SUBROUTINE h5tget_member_type_f

```

---

**FORTRAN interface: h5tcreate\_f**

```

SUBROUTINE h5tcreate_f(class, size, type_id, hdferr)
    IMPLICIT NONE
    INTEGER, INTENT(IN) :: class !Datatype class cna be one of
                                  !H5T_COMPOUND_F (6)
                                  !H5T_ENUM_F      (8)
                                  !H5T_OPAQUE_F    (9)
    INTEGER(SIZE_T), INTENT(IN) :: size !Size of the datatype

```

---

```
INTEGER(HID_T), INTENT(OUT) :: type_id !Datatype identifier
INTEGER, INTENT(OUT) :: hdferr          !Error code

END SUBROUTINE h5tcreate_f
```

---

**FORTRAN interface: h5tinsert\_f**

```
SUBROUTINE h5tinsert_f(type_id, name, offset, field_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  CHARACTER(LEN=*), INTENT(IN) :: name !Name of the field to insert
  INTEGER(SIZE_T), INTENT(IN) :: offset !Offset in memory structure of the
                                         !field to insert
  INTEGER(HID_T), INTENT(IN) :: field_id !datatype identifier of the new
                                         !member
  INTEGER, INTENT(OUT) :: hdferr          !Error code

END SUBROUTINE h5tinsert_f
```

---

**FORTRAN interface: h5tpack\_f**

```
SUBROUTINE h5tpack_f(type_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(OUT) :: hdferr          !Error code

END SUBROUTINE h5tpack_f
```

---

**FORTRAN interface: h5tenum\_create\_f**

```
SUBROUTINE h5tenum_create_f(parent_id, new_type_id, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: parent_id !Datatype identifier for
                                         !the base datatype
  INTEGER(HID_T), INTENT(OUT) :: new_type_id
                                         !datatype identifier for the
                                         !new enumeration datatype
  INTEGER, INTENT(OUT) :: hdferr          !Error code

END SUBROUTINE h5tenum_create_f
```

---

**FORTTRAN interface: h5tenum\_insert\_f**

```
SUBROUTINE h5tenum_insert_f(type_id, name, value, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  CHARACTER(LEN=*), INTENT(IN) :: name !Name of the new member
  INTEGER, INTENT(IN) :: value !value of the new member
  INTEGER, INTENT(OUT) :: hdferr !Error code

END SUBROUTINE h5tenum_insert_f
```

**FORTTRAN interface: h5tenum\_nameof\_f**

```
SUBROUTINE h5tenum_nameof_f(type_id, name, namelen, value, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  CHARACTER(LEN=*), INTENT(OUT) :: name !Name of the enumeration
  !datatype.
  INTEGER(SIZE_T), INTENT(IN) :: namelen !length of the name
  INTEGER, INTENT(IN) :: value !value of the enumeration
  !datatype.
  INTEGER, INTENT(OUT) :: hdferr !Error code

END SUBROUTINE h5tenum_nameof_f
```

**FORTTRAN interface: h5tenum\_valueof\_f**

```
SUBROUTINE h5tenum_valueof_f(type_id, name, value, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  CHARACTER(LEN=*), INTENT(IN) :: name !Name of the enumeration datatype.
  INTEGER, INTENT(OUT) :: value !value of the enumeration
  !datatype.
  INTEGER, INTENT(OUT) :: hdferr !Error code

END SUBROUTINE h5tenum_valueof_f
```

**FORTRAN interface: h5tget\_member\_value\_f**

```
SUBROUTINE h5tget_member_value_f(type_id, member_no, value, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  INTEGER, INTENT(IN) :: member_no      !Number of the enumeration datatype
                                          !member
  INTEGER, INTENT(OUT) :: value         !value of the enumeration
                                          !datatype.
  INTEGER, INTENT(OUT) :: hdferr       !Error code

END SUBROUTINE h5tget_member_value_f
```

---

**FORTRAN interface: h5tset\_tag\_f**

```
SUBROUTINE h5tset_tag_f(type_id, tag, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  CHARACTER(LEN=*), INTENT(IN) :: tag   !Unique ASCII string with which
                                          !the opaque datatype is to be
                                          !tagged
  INTEGER, INTENT(OUT) :: hdferr       !Error code

END SUBROUTINE h5tset_tag_f
```

---

**FORTRAN interface: h5tget\_tag\_f**

```
SUBROUTINE h5tget_tag_f(type_id, tag, taglen, hdferr)
  IMPLICIT NONE
  INTEGER(HID_T), INTENT(IN) :: type_id !Datatype identifier
  CHARACTER(LEN=*), INTENT(OUT) :: tag   !Unique ASCII string with which
                                          !the opaque datatype is to be
                                          !tagged
  INTEGER, INTENT(OUT) :: taglen        !length of tag
  INTEGER, INTENT(OUT) :: hdferr       !Error code

END SUBROUTINE h5tget_tag_f
```

---

# HDF5 Fortran90 Flags and Datatypes

## Fortran90 Datatypes

The Fortran90 HDF5 datatypes are listed in “HDF5 Predefined Datatypes” in the *HDF5 Reference Manual*.

## Fortran90 Flags

The Fortran90 HDF5 flags have the same meanings as the C flags defined in the *HDF5 Reference Manual* and the *HDF5 User's Guide*.

### File access flags

H5F\_ACC\_RDWR\_F

H5F\_ACC\_EXCL\_F

H5F\_SCOPE\_LOCAL\_F

H5F\_ACC\_RDONLY\_F

H5F\_ACC\_DEBUG\_F

H5F\_SCOPE\_GLOBAL\_F

H5F\_ACC\_TRUNC\_F

### Group management flags

H5G\_UNKNOWN\_F

H5G\_DATASET\_F

H5G\_LINK\_HARD\_F

H5G\_LINK\_F

H5G\_TYPE\_F

H5G\_LINK\_SOFT\_F

H5G\_GROUP\_F

H5G\_LINK\_ERROR\_F

### Dataset format flags

H5D\_COMPACT\_F

H5D\_CONTIGUOUS\_F

H5D\_CHUNKED\_F

### MPI IO data transfer flags

H5FD\_MPIO\_INDEPENDENT\_F

H5FD\_MPIO\_COLLECTIVE\_F

## Error flags

H5E_NONE_MAJOR_F	H5E_CACHE_F	H5E_STORAGE_F
H5E_ARGS_F	H5E_BTREE_F	H5E_PLIST_F
H5E_RESOURCE_F	H5E_SYM_F	H5E_ATTR_F
H5E_INTERNAL_F	H5E_HEAP_F	H5E_PLINE_F
H5E_FILE_F	H5E_OHDR_F	H5E_EFL_F
H5E_IO_F	H5E_DATATYPE_F	H5E_REFERENCE_F
H5E_FUNC_F	H5E_DATASPACE_F	H5E_VFL_F
H5E_ATOM_F	H5E_DATASET_F	H5E_TBBT_F

## Object identifier flags

H5I_FILE_F	H5I_DATASPACE_F	H5I_BADID_F
H5I_GROUP_F	H5I_DATASET_F	
H5I_DATATYPE_F	H5I_ATTR_F	

## Property list flags

H5P_FILE_CREATE_F	H5P_DATASET_CREATE_F	H5P_MOUNT_F
H5P_FILE_ACCESS_F	H5P_DATASET_XFER_F	H5P_DEFAULT_F

## Reference pointer flags

H5R_OBJECT_F	H5R_DATASET_REGION_F
--------------	----------------------

## Dataspace flags

H5S_SCALAR_F	H5S_SELECT_SET_F	H5S_UNLIMITED_F
H5S_SIMPLE_F	H5S_SELECT_OR_F	H5S_ALL_F

**Datatype flags**

H5T_NO_CLASS_F	H5T_ORDER_LE_F	H5T_NORM_IMPLIED_F
H5T_INTEGER_F	H5T_ORDER_BE_F	H5T_NORM_MSBSET_F
H5T_FLOAT_F	H5T_ORDER_VAX_F	H5T_NORM_NONE_F
H5T_TIME_F	H5T_PAD_ZERO_F	H5T_CSET_ASCII_F
H5T_STRING_F	H5T_PAD_ONE_F	H5T_STR_NULLTERM_F
H5T_BITFIELD_F	H5T_PAD_BACKGROUND_F	H5T_STR_NULLPAD_F
H5T_OPAQUE_F	H5T_PAD_ERROR_F	H5T_STR_SPACEPAD_F
H5T_COMPOUND_F	H5T_SGN_NONE_F	H5T_STR_ERROR_F
H5T_REFERENCE_F	H5T_SGN_2_F	
H5T_ENUM_F	H5T_SGN_ERROR_F	

---



