# Package 'rhdf5'

October 17, 2020

**Type** Package

**Title** R Interface to HDF5

**Version** 2.32.4

**Description** This package provides an interface between HDF5 and R. HDF5's main features are the ability to store and access very large and/or complex datasets and a wide variety of metadata on mass storage (disk) through a completely portable file format. The rhdf5 package is thus suited for the exchange of large and/or complex datasets between R and other software package, and for letting R applications work on datasets that are larger than the available RAM.

**License** Artistic-2.0

**URL** https://github.com/grimbough/rhdf5

**BugReports** https://github.com/grimbough/rhdf5/issues

**LazyLoad** true

**VignetteBuilder** knitr

**Imports** Rhdf5lib (>= 1.3.2)

**Depends** R (>= 3.5.0), methods

**Suggests** bit64, BiocStyle, knitr, rmarkdown, testthat, microbenchmark, dplyr, ggplot2

**LinkingTo** Rhdf5lib

**SystemRequirements** GNU make

**biocViews** Infrastructure, DataImport

**Encoding** UTF-8

**RoxygenNote** 6.1.1

**git_url** https://git.bioconductor.org/packages/rhdf5

**git_branch** RELEASE_3_11

**git_last_commit** 5316244

**git_last_commit_date** 2020-10-05

**Date/Publication** 2020-10-16

**Author** Bernd Fischer [aut],
Mike Smith [aut, cre] (<https://orcid.org/0000-0002-7800-3848>),
Gregoire Pau [aut],
Martin Morgan [ctb],
Daniel van Twisk [ctb]

**Maintainer** Mike Smith <mike.smith@embl.de>

# R topics documented:

h5closeAll . . . 2
h5const . . . 3
h5createAttribute . . . 4
h5createDataset . . . 5
h5createFile . . . 7
h5createGroup . . . 8
h5delete . . . 9
h5errorHandling . . . 10
H5IdComponent-class . . . 11
h5listIdentifier . . . 12
h5ls . . . 13
h5save . . . 14
h5set_extent . . . 15
h5testFileLocking . . . 16
h5version . . . 18
h5write . . . 18
HDF5 Attribute Interface . . . 22
HDF5 Dataset Access Property List Interface . . . 24
HDF5 Dataset Create Property List Interface . . . 25
HDF5 Dataset Interface . . . 27
HDF5 Dataspace Interface . . . 29
HDF5 Datatype Interface . . . 32
HDF5 File Interface . . . 33
HDF5 General Library Functions . . . 34
HDF5 Group Interface . . . 35
HDF5 Identifier Interface . . . 37
HDF5 Link Create Property List Interface . . . 38
HDF5 Link Interface . . . 39
HDF5 Object Interface . . . 41
HDF5 Property List Interface . . . 42
rhdf5 . . . 44
[-methods . . . 45
[<–methods . . . 46
$-methods . . . 46
$<–methods . . . 46
&-methods . . . 47

**Index** **48**

---

`h5closeAll` *Close all open HDF5 handles*

---

## Description

Occasionally references to HDF5 files, groups, datasets etc can be created and not closed correctly. This function identifies all open handles and closes them. It replaces the functionality previously supplied by `H5close`.

## Usage

```
h5closeAll()
```

## Author(s)

Mike Smith

## Examples

```
## create an empty file and then re-open it
h5createFile("ex_h5closeAll.h5")
H5Fopen("ex_h5closeAll.h5")

## list all open identifiers
h5listIdentifier()

## close all open identifiers and verify
h5closeAll()
h5listIdentifier()
```

---

# h5const *HDF5 library constants.*

---

## Description

Access to HDF5 constants.

## Usage

```
h5const     (type = "")
h5default   (type = "")
h5constType ()
```

## Arguments

| | |
|---|---|
| type | A character name of a group of constants. |

## Details

These functions provide a list of HDF5 constants that are defined in the R package. `h5constType` provides a list of group names and `h5const` gives the constants defined within a group. `h5default` gives the default choice for each group.

## Value

A character vector with names of HDF5 constants or groups.

## Author(s)

Bernd Fischer

## References

https://portal.hdfgroup.org/display/HDF5

## See Also

rhdf5

## Examples

```
h5constType()[1]
h5const(h5constType()[1])
```

---

`h5createAttribute` *Create HDF5 attribute*

---

## Description

R function to create an HDF5 attribute and defining its dimensionality.

## Usage

```
h5createAttribute (obj, attr, dims, maxdims = dims, file,
                   storage.mode = "double", H5type = NULL, size=NULL,
                   native = FALSE)
```

## Arguments

| | |
|---|---|
| `obj` | The name (character) of the object the attribute will be attatched to. For advanced programmers it is possible to provide an object of class `H5IdComponent` representing a H5 object identifier (file, group, dataset). See `H5Fcreate`, `H5Fopen`, `H5Gcreate`, `H5Gopen`, `H5Dcreate`, `H5Dopen` to create an object of this kind. |
| `file` | The filename (character) of the file in which the dataset will be located. For advanced programmers it is possible to provide an object of class `H5IdComponent` representig an H5 location identifier. See `H5Fcreate`, `H5Fopen`, `H5Gcreate`, `H5Gopen` to create an object of this kind. The `file` argument is not required, if the argument `obj` is of type `H5IdComponent`. |
| `attr` | Name of the attribute to be created. |
| `dims` | The dimension of the attribute. |
| `maxdims` | The maximum extension of the attribute. |
| `storage.mode` | The storage mode of the data to be written. Can be obtained by `storage.mode(mydata)`. |
| `H5type` | Advanced programmers can specify the datatype of the dataset within the file. See `h5const("H5T")` for a list of available datatypes. If `H5type` is specified the argument `storage.mode` is ignored. It is recommended to use `storage.mode` |
| `size` | For `storage.mode='character'` the maximum string length has to be specified. HDF5 then stores the string as fixed length character vectors. Together with compression, this should be efficient. |
| `native` | An object of class `logical`. If TRUE, array-like objects are treated as stored in HDF5 row-major rather than R column-major orientation. Using `native = TRUE` increases HDF5 file portability between programming languages. A file written with `native = TRUE` should also be read with `native = TRUE`. |

### Details

Creates a new attribute and attaches it to an existing HDF5 object. The function will fail, if the file doesn't exist or if there exists already another attribute with the same name for this object.

You can use h5writeAttribute immediately. It will create the attribute for you.

### Value

Returns TRUE is attribute was created successfully and FALSE otherwise.

### Author(s)

Bernd Fischer

### References

https://portal.hdfgroup.org/display/HDF5

### See Also

h5createFile, h5createGroup, h5createDataset, h5read, h5write, rhdf5

### Examples

```
h5createFile("ex_createAttribute.h5")
h5write(1:1, "ex_createAttribute.h5","A")
fid <- H5Fopen("ex_createAttribute.h5")
did <- H5Dopen(fid, "A")
h5createAttribute (did, "time", c(1,10))
H5Dclose(did)
H5Fclose(fid)
```

---

## h5createDataset *Create HDF5 dataset*

---

### Description

R function to create an HDF5 dataset and defining its dimensionality and compression behaviour.

### Usage

```
h5createDataset (file, dataset,
dims, maxdims = dims,
storage.mode = "double", H5type = NULL,
size = NULL, chunk = dims, level = 6,
fillValue, showWarnings, native = FALSE)
```

## Arguments

| | |
|---|---|
| file | The filename (character) of the file in which the dataset will be located. For advanced programmers it is possible to provide an object of class H5IdComponent representing a H5 location identifier (file or group). See H5Fcreate, H5Fopen, H5Gcreate, H5Gopen to create an object of this kind. |
| dataset | Name of the dataset to be created. The name can contain group names, e.g. 'group/dataset', but the function will fail, if the group does not yet exist. |
| dims | The dimensions of the array as they will appear in the file. Note, the dimensions will appear in inverted order when viewing the file with a C-programm (e.g. HDFView), because the fastest changing dimension in R is the first one, whereas the fastest changing dimension in C is the last one. |
| maxdims | The maximum extension of the array. Use H5Sunlimited() to indicate an extensible dimension. |
| storage.mode | The storage mode of the data to be written. Can be obtained by storage.mode(mydata). |
| H5type | Advanced programmers can specify the datatype of the dataset within the file. See h5const("H5T") for a list of available datatypes. If H5type is specified the argument storage.mode is ignored. It is recommended to use storage.mode |
| size | For storage.mode='character' the maximum string length has to be specified. rhdf5 writes null padded strings by dafault, thus the value provided here should be the length of the longest string. HDF5 then stores the string as fixed length character vectors. Together with compression, this should be efficient. |
| chunk | The chunk size used to store the dataset. It is an integer vector of the same length as dims. This argument is usually set together with a compression property (argument level). |
| level | The compression level used. An integer value between 0 (no compression) and 9 (highest and slowest compression). |
| fillValue | Standard value for filling the dataset. The storage.mode of value has to be convertable to the dataset type by HDF5. |
| showWarnings | Deprecated. Use suppressMessages and suppressWarnings to limit messages printed to screen. |
| native | An object of class logical. If TRUE, array-like objects are treated as stored in HDF5 row-major rather than R column-major orientation. Using native = TRUE increases HDF5 file portability between programming languages. A file written with native = TRUE should also be read with native = TRUE. |

## Details

Creates a new dataset. in an existing HDF5 file. The function will fail, if the file doesn't exist or if there exists already another dataset with the same name within the specified file.

## Value

Returns TRUE is dataset was created successfully and FALSE otherwise.

## Author(s)

Bernd Fischer

## References

https://portal.hdfgroup.org/display/HDF5

## See Also

h5createFile, h5createGroup, h5read, h5write, rhdf5

## Examples

```
h5createFile("ex_createDataset.h5")

# create dataset with compression
h5createDataset("ex_createDataset.h5", "A", c(5,8), storage.mode = "integer", chunk=c(5,1), level=6)

# create dataset without compression
h5createDataset("ex_createDataset.h5", "B", c(5,8), storage.mode = "integer")
h5createDataset("ex_createDataset.h5", "C", c(5,8), storage.mode = "double")

# create a dataset of strings & define size based on longest string
ex_strings <- c('long', 'longer', 'longest')
h5createDataset("ex_createDataset.h5", "D",
    storage.mode = "character", chunk = 3, level = 6,
    dims = length(ex_strings), size = max(nchar(ex_strings)))


# write data to dataset
h5write(matrix(1:40,nr=5,nc=8), file="ex_createDataset.h5", name="A")
# write second column
h5write(matrix(1:5,nr=5,nc=1), file="ex_createDataset.h5", name="B", index=list(NULL,2))
# write character vector
h5write(ex_strings, file = "ex_createDataset.h5", name = "D")

h5dump("ex_createDataset.h5")
```

---

# h5createFile *Create HDF5 file*

---

## Description

R function to create an empty HDF5 file.

## Usage

```
h5createFile (file)
```

## Arguments

| | |
|---|---|
| file | The filename of the HDF5 file. |

## Details

Creates an empty HDF5 file.

## Value

Returns TRUE is file was created successfully and FALSE otherwise.

## Author(s)

Bernd Fischer

## References

https://portal.hdfgroup.org/display/HDF5

## See Also

h5createGroup, h5createDataset, h5read, h5write, rhdf5

## Examples

```
h5createFile("ex_createFile.h5")

# create groups
h5createGroup("ex_createFile.h5","foo")
h5createGroup("ex_createFile.h5","foo/foobaa")

h5ls("ex_createFile.h5")
```

---

# h5createGroup *Create HDF5 group*

---

## Description

Creates a group within an HDF5 file.

## Usage

```
h5createGroup (file, group)
```

## Arguments

| | |
|---|---|
| file | The filename (character) of the file in which the dataset will be located. For advanced programmers it is possible to provide an object of class H5IdComponent representing a H5 location identifier (file or group). See H5Fcreate, H5Fopen, H5Gcreate, H5Gopen to create an object of this kind. |
| group | The name of the new group. The name can contain a hierarchy of groupnames, e.g. 'group1/group2/newgroup', but the function will fail if the top level group do not exists. |

## Details

Creates a new group within an HDF5 file.

## Value

Returns TRUE is group was created successfully and FALSE otherwise.

### Author(s)

Bernd Fischer

### References

https://portal.hdfgroup.org/display/HDF5

### See Also

h5createFile, h5createDataset, h5read, h5write, rhdf5

### Examples

```
h5createFile("ex_createGroup.h5")

# create groups
h5createGroup("ex_createGroup.h5","foo")
h5createGroup("ex_createGroup.h5","foo/foobaa")

h5ls("ex_createGroup.h5")
```

---

## h5delete *Delete contents of HDF5 file*

---

### Description

Deletes the specfied group or dataset from within an HDF5 file.

### Usage

```
h5delete( file, name )
```

### Arguments

| | |
|---|---|
| file | The filename (character) of the file in which the object is located. |
| name | The names of the object to be deleted. |

### Author(s)

Mike Smith

`h5errorHandling` *handling of HDF5 error messages*

## Description

Sets the options for handling HDF5 error messages.

## Usage

```
h5errorHandling (type="normal")
```

## Arguments

| | |
|---|---|
| `type` | 'normal' (default) shows a one line error message in R. 'verbose' shows the whole HDF5 error message. 'suppress' suppresses the HDF5 error messages completely. |

## Details

Sets the options for HDF5 error hanlding.

## Value

Returns 0 if options are set successfully.

## Author(s)

Bernd Fischer

## References

https://portal.hdfgroup.org/display/HDF5

## See Also

rhdf5

## Examples

```
h5errorHandling("normal")
```

H5IdComponent-class *Class "H5IdComponent"*

## Description

A class representing a HDF5 identifier handle. HDF5 identifiers represent open files, groups, datasets, dataspaces, attributes, and datatypes.

## Objects from the Class

Objects can be created by calls of H5Fcreate, H5Fopen, / H5Gcreate, H5Gopen, / H5Dcreate, H5Dopen, \ H5Dget_space, H5Screate_simple, \ H5Acreate, H5Aopen.

## Slots

ID: Object of class "integer". Contains the handle of C-type hid_t.

**native** An object of class logical. If TRUE, array-like objects are treated as stored in HDF5 row-major rather than R column-major orientation. Using native = TRUE increases HDF5 file portability between programming languages. A file written with native = TRUE should also be read with native = TRUE.

## Methods

**show** signature(object = "H5IdComponent"): Shows the filename.

## Author(s)

Bernd Fischer

## References

https://portal.hdfgroup.org/display/HDF5

## See Also

rhdf5

## Examples

```
showClass("H5IdComponent")
```

h5listIdentifier *list all valid H5 identifier.*

## Description

A list of all valid H5 identifier. H5 objects should be closed after usage to release resources.

## Usage

```
h5listIdentifier()
h5validObjects(native = FALSE)
```

## Arguments

| | |
|---|---|
| native | An object of class logical. If TRUE, array-like objects are treated as stored in HDF5 row-major rather than R column-major orientation. Using native = TRUE increases HDF5 file portability between programming languages. A file written with native = TRUE should also be read with native = TRUE. |

## Value

h5validObjects returns a list of H5IdComponent objects. h5listIdentifier prints the valid identifiers on screen and returns NULL.

## Author(s)

Bernd Fischer

## References

https://portal.hdfgroup.org/display/HDF5

## See Also

rhdf5

## Examples

```
h5createFile("ex_list_identifier.h5")

# create groups
h5createGroup("ex_list_identifier.h5","foo")

h5listIdentifier()
h5validObjects()
```

h5ls *List the content of an HDF5 file.*

## Description

Lists the content of an HDF5 file.

## Usage

```
h5ls   (file,
        recursive = TRUE,
all = FALSE,
datasetinfo = TRUE,
index_type = h5default("H5_INDEX"),
order = h5default("H5_ITER"), native = FALSE)
h5dump (file,
        recursive = TRUE,
load = TRUE,
all = FALSE,
        index_type = h5default("H5_INDEX"),
        order = h5default("H5_ITER"), ..., native = FALSE)
```

## Arguments

| | |
|---|---|
| file | The filename (character) of the file in which the dataset will be located. For advanced programmers it is possible to provide an object of class H5IdComponent representing a H5 location identifier (file or group). See H5Fcreate, H5Fopen, H5Gcreate, H5Gopen to create an object of this kind. |
| recursive | If TRUE, the content of the whole group hierarchy is listed. If FALSE, Only the content of the main group is shown. If a positive integer is provided this indicates the maximum level of the hierarchy that is shown. |
| all | If TRUE, a longer list of information on each entry is provided. |
| datasetinfo | If FALSE, datatype and dimensionality information is not provided. This can speed up the content listing for large files. |
| index_type | See h5const("H5_INDEX") for possible arguments. |
| order | See h5const("H5_ITER") for possible arguments. |
| load | If TRUE the datasets are read in, not only the header information. Note, that this can cause memory problems for very large files. In this case choose load=FALSE and load the datasets successively. |
| native | An object of class logical. If TRUE, array-like objects are treated as stored in HDF5 row-major rather than R column-major orientation. Using native = TRUE increases HDF5 file portability between programming languages. A file written with native = TRUE should also be read with native = TRUE. |
| ... | Arguments passed to h5read |

## Details

h5ls lists the content of an HDF5 file including group structure and datasets. It returns the content as a data.frame. You can use h5dump(file="myfile.h5",load=FALSE) to obtain the dataset information in a hierarchical list structure. Usually the datasets are loaded individually with h5read, but you have the possibility to load the complete content of an HDF5 file with h5dump

## Value

h5ls returns a data.frame with the file content.

h5dump returns a hierarchical list structure representing the HDF5 group hierarchy. It either returns the datasets within the list structure (load=TRUE) or it returns a data.frame for each datset with the dataset header information load=FALSE.

## Author(s)

Bernd Fischer

## References

https://portal.hdfgroup.org/display/HDF5

## See Also

h5read, h5write, rhdf5

## Examples

```
h5createFile("ex_ls_dump.h5")

# create groups
h5createGroup("ex_ls_dump.h5","foo")
h5createGroup("ex_ls_dump.h5","foo/foobaa")

# write a matrix
B = array(seq(0.1,2.0,by=0.1),dim=c(5,2,2))
attr(B, "scale") <- "liter"
h5write(B, "ex_ls_dump.h5","foo/B")

# list content of hdf5 file
h5ls("ex_ls_dump.h5",all=TRUE)
h5dump("ex_ls_dump.h5")
```

---

# h5save *Saves a series of objects to an HDF5 file.*

---

## Description

Saves a number of R objects to an HDF5 file.

## Usage

```
h5save(..., file, name = NULL, createnewfile = TRUE, native = FALSE)
```

### Arguments

| | |
|---|---|
| ... | The objects to be saved. |
| file | The filename (character) of the file in which the dataset will be located. For advanced programmers it is possible to provide an object of class H5IdComponent representing a H5 location identifier (file or group). See H5Fcreate, H5Fopen, H5Gcreate, H5Gopen to create an object of this kind. |
| name | A character vector of names for the datasets. The length of the name vector should match the number of objects. |
| createnewfile | If TRUE, a new file will be created if necessary. |
| native | An object of class logical. If TRUE, array-like objects are treated as stored in HDF5 row-major rather than R column-major orientation. Using native = TRUE increases HDF5 file portability between programming languages. A file written with native = TRUE should also be read with native = TRUE. |

### Details

The objects will be saved to the HDF5 file. If the file does not exists it will be created. The data can be read again by either h5dump or individually for each dataset by h5read.

### Value

Nothing returned.

### Author(s)

Bernd Fischer

### References

https://portal.hdfgroup.org/display/HDF5

### See Also

h5ls, h5write, rhdf5

### Examples

```
A = 1:7;  B = 1:18; D = seq(0,1,by=0.1)
h5save(A, B, D, file="ex_save.h5")
h5dump("ex_save.h5")
```

---

## h5set_extent *Set a new dataset extension*

---

### Description

Set a new dataset extension to an existing dataset in an HDF5 file

### Usage

```
h5set_extent(file, dataset, dims, native = FALSE)
```

### Arguments

| | |
|---|---|
| file | The filename (character) of the file in which the dataset will be located. For advanced programmers it is possible to provide an object of class H5IdComponent representing a H5 location identifier (file or group). See H5Fcreate, H5Fopen, H5Gcreate, H5Gopen to create an object of this kind. |
| dataset | The name of the dataset in the HDF5 file, or an object of class H5IdComponent representing a H5 dataset identifier. See H5Dcreate, or H5Dopen to create an object of this kind. |
| dims | The dimensions of the array as they will appear in the file. Note, the dimensions will appear in inverted order when viewing the file with a C-programm (e.g. HDFView), because the fastest changing dimension in R is the first one, whereas the fastest changing dimension in C is the last one. |
| native | An object of class logical. If TRUE, array-like objects are treated as stored in HDF5 row-major rather than R column-major orientation. Using native = TRUE increases HDF5 file portability between programming languages. A file written with native = TRUE should also be read with native = TRUE. |

### Value

Returns 0 if the dimension of the dataset was changed successfully and a negative value otherwise.

### Author(s)

Bernd Fischer

### References

https://portal.hdfgroup.org/display/HDF5

### See Also

rhdf5

### Examples

```
tmpfile <- tempfile()
h5createFile(file=tmpfile)
h5createDataset(tmpfile, "A", c(10,12), c(20,24))
h5ls(tmpfile, all=TRUE)[c("dim", "maxdim")]
h5set_extent(tmpfile, "A", c(20,24))
h5ls(tmpfile, all=TRUE)[c("dim", "maxdim")]
```

---

## h5testFileLocking *Test and set file locking for HDF5*

---

### Description

HDF5 1.10 uses file locking by default. On some file systems this is not available, and the HDF5 library will throw an error if the user attempts to create or access a file located on such a file system. These functions help identify if file locking is available without throwing an error, and allow the locking to be disabled for the duration of the R session if needed.

## Usage

```
h5testFileLocking(location)
h5disableFileLocking()
h5enableFileLocking()
```

## Arguments

| | |
|---|---|
| location | The name of a directory or file to test. If an existing directory is provided a temporary file will be created in this folder. If non-existant location is provided a file with the name will be created, tested for file locking, and then removed. Providing an existing file will result in an error. |

## Details

h5testFileLocking will create a temporary file and then attempt to apply a file lock using the appropriate function within the HDF5 library. The success or failure of the locking is then recorded and the temporary file removed. Even relatively low level functions such as H5Fcreate will fail inelegantly if file locking fails.

h5disableFileLocking will set the environment variable RHDF5_USE_FILE_LOCKING=FALSE, which is the recommended was to disable this behaviour if file locking is not supported. This will only persist within the current R session. You can set the environment variable outside of R if this is a more general issue on your system.

h5enableFileLocking will unset the RHDF5_USE_FILE_LOCKING.

More discussion of HDF5's use of file locking can be found online e.g. https://forum.hdfgroup.org/t/hdf5-1-10-0-and-flock/3761/4 or https://forum.hdfgroup.org/t/hdf5-files-on-nfs/3985/5

## Value

h5testFileLocking returns TRUE if a file can be successfully locked at the specified location, or FALSE otherwise.

h5disableFileLocking and h5enableFileLocking set are called for the side effect of setting or unsetting the environment variable HDF5_USE_FILE_LOCKING and do not return anything.

## Author(s)

Mike Smith

## Examples

```
## either a file name or directory can be tested
file <- tempfile()
dir <- tempdir()

h5testFileLocking(dir)
h5testFileLocking(file)

## we can check for file locking, and disable if needed
if( !h5testFileLocking(dir) ) {
  h5disableFileLocking()
}
```

## h5version *rhdf5 and libhdf5 version numbers*

### Description

Returns the version number of the Bioconductor package rhdf5 and the C-library libhdf5.

### Usage

```
h5version()
```

### Value

A list of major, minor and release number.

### Author(s)

Bernd Fischer

### References

https://portal.hdfgroup.org/display/HDF5

### See Also

rhdf5

### Examples

```
h5version()
```

## h5write *Reads and write object in HDF5 files*

### Description

Reads and writes objects in HDF5 files. This function can be used to read and write either full arrays/vectors or subarrays (hyperslabs) within an existing dataset.

### Usage

```
h5read                    (file, name, index=NULL,
                           start=NULL, stride=NULL, block=NULL,
                           count=NULL, compoundAsDataFrame = TRUE,
                           callGeneric = TRUE,
                           read.attributes = FALSE, drop = FALSE,
                           ..., native = FALSE)
h5readAttributes          (file, name, native = FALSE)
h5write                   (obj, file, name, ...)
h5write.default           (obj, file, name,
```

```
                              createnewfile = TRUE,
                              write.attributes = FALSE, ...,
                              native = FALSE)
h5writeDataset               (obj, h5loc, name, ...)
h5writeDataset.data.frame    (obj, h5loc, name, level=6, chunk,
                              DataFrameAsCompound = TRUE)
h5writeDataset.list          (obj, h5loc, name, level=6)
h5writeDataset.matrix        (...)
h5writeDataset.integer       (...)
h5writeDataset.double        (...)
h5writeDataset.logical       (...)
h5writeDataset.character     (...)
h5writeDataset.array         (obj, h5loc, name, index = NULL,
                              start=NULL, stride=NULL, block=NULL, count=NULL,
                              size=NULL, level=6)
h5writeAttribute             (attr, h5obj, name, ...)
h5writeAttribute.matrix      (...)
h5writeAttribute.integer     (...)
h5writeAttribute.double      (...)
h5writeAttribute.logical     (...)
h5writeAttribute.character   (...)
h5writeAttribute.array       (attr, h5obj, name, size)
```

## Arguments

| | |
|---|---|
| obj | The R object to be written. |
| attr | The R object to be written as an HDF5 attribute. |
| file | The filename (character) of the file in which the dataset will be located. For advanced programmers it is possible to provide an object of class H5IdComponent representing a H5 location identifier (file or group). See H5Fcreate, H5Fopen, H5Gcreate, H5Gopen to create an object of this kind. |
| h5loc | An object of class H5IdComponent representing a H5 location identifier (file or group). See H5Fcreate, H5Fopen, H5Gcreate, H5Gopen to create an object of this kind. |
| h5obj | An object of class H5IdComponent representing a H5 object identifier (file, group, or dataset). See H5Fcreate, H5Fopen, H5Gcreate, H5Gopen, H5Dcreate, or H5Dopen to create an object of this kind. |
| name | The name of the dataset in the HDF5 file. The name of the attribute for hwriteAttribute. |
| index | List of indices for subsetting. The length of the list has to agree with the dimensional extension of the HDF5 array. Each list element is an integer vector of indices. A list element equal to NULL choses all indices in this dimension. Counting is R-style 1-based. |
| start | The start coordinate of a hyperslab (similar to subsetting in R). Counting is R-style 1-based. This argument is ignored, if index is not NULL. |
| stride | The stride of the hypercube. Read the introduction http://ftp.hdfgroup.org/HDF5/Tutor/phypecont.html before using this argument. R behaves like Fortran in this example. This argument is ignored, if index is not NULL. |
| block | The block size of the hyperslab. Read the introduction http://ftp.hdfgroup.org/HDF5/Tutor/phypecont.html before using this argument. R behaves like Fortran in this example. This argument is ignored, if index is not NULL. |

count — The number of blocks to be written. This argument is ignored, if index is not NULL.

level — The compression level. An integer value between 0 (no compression) and 9 (highest and slowest compression). Only used, if the dataset does not yet exist. See h5createDataset to create an dataset.

chunk — Specifies the number of items to be include in an HDF5 chunk. When writing a data.frame this represents the number of rows to be included in a chunk. If left unspecified the defaults is the smaller of: the total number of rows or the number of rows that fit within 4GB of memory.

native — An object of class logical. If TRUE, array-like objects are treated as stored in HDF5 row-major rather than R column-major orientation. Using native = TRUE increases HDF5 file portability between programming languages. A file written with native = TRUE should also be read with native = TRUE.

compoundAsDataFrame — If true, a compound datatype will be coerced to a data.frame. This is not possible, if the dataset is multi-dimensional. Otherwise the compound datatype will be returned as a list. Nested compound data types will be returned as a nested list.

DataFrameAsCompound — If true, a data.frame will be saved as a compound data type. Otherwise it is saved like a list. The advantage of saving a data.frame as a compound data type is that it can be read as a table from python or with a struct-type from C. The disadvantage is that the data has to be rearranged on disk and thus can slow down I/O. If fast reading is required, DataFrameAsCompound=FALSE is recommended.

callGeneric — If TRUE a generic function h5read.classname will be called if it exists depending on the dataset's class attribute within the HDF5 file. This function can be used to convert the standard output of h5read depending on the class attribute. Note that h5read is not a S3 generic function. Dispatching is done based on the HDF5 attribute after the standard h5read function.

size — The length of string data type. Variable lengt strings are not yet supported.

createnewfile — If TRUE, a new file will be created if necessary.

read.attributes — (logical) If TRUE, the HDF5 attributes are read and attached to the respective R object.

drop — (logical) If TRUE, the HDF5 object is read as a vector with NULL dim attributes.

write.attributes — (logical) If TRUE, all R-attributes attached to the object obj are written to the HDF5 file.

... — Further arguments passed to H5Dread.

## Details

Read/writes an R object from/to an HDF5 file. If neither of the arguments start,stride,block,count is specified, the dataset has the same dimension in the HDF5 file and in memory. If the dataset already exists in the HDF5 file, one can read/write subarrays, so called hyperslabs from/to the HDF5 file. The arguments start,stride,block,count define the subset of the dataset in the HDF5 file that is to be read/written. See these introductions to hyperslabs: https://support.hdfgroup.org/HDF5/Tutor/selectsimple.html, https://support.hdfgroup.org/HDF5/Tutor/select.

html and http://ftp.hdfgroup.org/HDF5/Tutor/phypecont.html. Please note that in R the first dimension is the fastest changing dimension.

When viewing the HDF5 datasets with any C-program (e.g. HDFView), the order of dimensions is inverted. In the R interface counting starts with 1, whereas in the C-programs (e.g. HDFView) counting starts with 0.

## Value

h5read returns an array with the data read.

h5readAttributes returns a list of all HDF5 attributes of object name.

h5write returns 0 if successful.

## Author(s)

Bernd Fischer

## References

https://portal.hdfgroup.org/display/HDF5

## See Also

h5ls, h5createFile, h5createDataset, rhdf5

## Examples

```
h5createFile("ex_hdf5file.h5")

# write a matrix
B = array(seq(0.1,2.0,by=0.1),dim=c(5,2,2))
attr(B, "scale") <- "liter"
h5write(B, "ex_hdf5file.h5","B")

# read a matrix
E = h5read("ex_hdf5file.h5","B")

# write and read submatrix
h5createDataset("ex_hdf5file.h5", "S", c(5,8), storage.mode = "integer", chunk=c(5,1), level=7)
h5write(matrix(1:5,nr=5,nc=1), file="ex_hdf5file.h5", name="S", index=list(NULL,1))
h5read("ex_hdf5file.h5", "S")
h5read("ex_hdf5file.h5", "S", index=list(NULL,2:3))

# list content of hdf5 file
h5ls("ex_hdf5file.h5")
```

HDF5 Attribute Interface

*HDF5 Attribute Interface*

## Description

These functions create and manipulate attributes and information about attributes.

## Usage

```
H5Acreate       (h5obj, name, dtype_id, h5space)
H5Aclose        (h5attribute)
H5Adelete       (h5obj, name)
H5Aexists       (h5obj, name)
H5Aget_name     (h5attribute)
H5Aget_space    (h5attribute)
H5Aget_type     (h5attribute)
H5Aopen         (h5obj, name)
H5Aopen_by_idx  (h5obj, n, objname = ".", index_type = h5default("H5_INDEX"),
                 order = h5default("H5_ITER"))
H5Aopen_by_name (h5obj, objname = ".", name)
H5Aread         (h5attribute, buf = NULL)
H5Awrite        (h5attribute, buf)
```

## Arguments

| | |
|---|---|
| h5obj | An object of class H5IdComponent representing a H5 object identifier (file, group, or dataset). See H5Fcreate, H5Fopen, H5Gcreate, H5Gopen, H5Dcreate, or H5Dopen to create an object of this kind. |
| name | The name of the attribute (character). |
| dtype_id | A character name of a datatype. See h5const("H5T") for possible datatypes. Can also be an integer representing an HDF5 datatype. Only simple datatypes are allowed for atttributes. |
| h5space | An object of class H5IdComponent representing a H5 dataspace. See H5Dget_space, H5Screate_simple, H5Screate to create an object of this kind. |
| h5attribute | An object of class H5IdComponent represnting a H5 attribute as created by H5Acreate or H5Aopen |
| n | Opens attribute number n in the given order and index. The first attribute is opened with n=0. |
| objname | The name of the object the attribute belongs to. |
| index_type | See h5const("H5_INDEX") for possible arguments. |
| order | See h5const("H5_ITER") for possible arguments. |
| buf | Reading and writing buffer containing the data to written/read. When using the buffer for reading, the buffer size has to fit the size of the memory space h5spaceMem. No extra memory will be allocated for the data. A pointer to the same data is returned. |

## Details

Interface to the HDF5 C-library libhdf5. See https://portal.hdfgroup.org/display/HDF5/Attributes for further details.

## Value

H5Acreate, H5Aopen, H5Aopen_by_name, H5Aopen_by_idx return an object of class H5IdComponent representing a H5 attribute identifier.

H5Aget_space returns an object of class H5IdComponent representing a H5 dataspace identifier.

H5Aread returns an array with the read data.

The other functions return the standard return value from their respective C-functions.

## Author(s)

Bernd Fischer

## References

https://portal.hdfgroup.org/display/HDF5

## See Also

rhdf5

## Examples

```
# create a file and write something
h5createFile("ex_H5A.h5")
h5write(1:15, "ex_H5A.h5","A")

# write an attribute 'unit' to 'A'
fid <- H5Fopen("ex_H5A.h5")
did <- H5Dopen(fid, "A")
sid <- H5Screate_simple(c(1,1))
tid <- H5Tcopy("H5T_C_S1")

H5Tset_size(tid, 10L)
aid <- H5Acreate(did, "unit", tid, sid)
aid
H5Awrite(aid, "liter")
H5Aclose(aid)
H5Sclose(sid)
H5Aexists(did, "unit")
H5Dclose(did)
H5Fclose(fid)
h5dump("ex_H5A.h5")
```

HDF5 Dataset Access Property List Interface

*HDF5 Dataset Access Property List Interface*

## Description

The functions, macros, and subroutines listed here are used to manipulate dataset access property list objects in various ways, including to reset property values. With the use of property lists, HDF5 functions have been implemented and can be used in applications with many fewer parameters than would be required without property lists.

## Usage

```
H5Pset_chunk_cache              ( h5plist, rdcc_nslots, rdcc_nbytes, rdcc_w0 )
```

## Arguments

| | |
|---|---|
| h5plist | An object of class H5IdComponent representing a H5 property list identifier of class H5P_DATASET_ACCESS. See H5Pcreate or H5Pcopy to create an object of this kind. |
| rdcc_nslots | An integer. The number of chunk slots in the raw data chunk cache for this dataset. |
| rdcc_nbytes | An integer. The total size of the raw data chunk cache for this dataset. |
| rdcc_w0 | double. The chunk preemption policy for this dataset. |

## Details

Interface to the HDF5 C-library libhdf5. See https://portal.hdfgroup.org/display/HDF5/Dataset+Access+Properties for further details. See H5P for documentation of more functions to manupulate property lists of other property list classes.

## Value

The functions return the standard return value from their respective C-functions.

## Author(s)

Bernd Fischer

## References

https://portal.hdfgroup.org/display/HDF5

## See Also

rhdf5, H5P

## Examples

```
pid <- H5Pcreate("H5P_DATASET_ACCESS")
H5Pset_chunk_cache( pid, 100, 10000, 0.5)
H5Pclose(pid)
```

---

HDF5 Dataset Create Property List Interface

*HDF5 Dataset Create Property List Interface*

---

## Description

The functions, macros, and subroutines listed here are used to manipulate dataset creation property list objects in various ways, including to reset property values. With the use of property lists, HDF5 functions have been implemented and can be used in applications with many fewer parameters than would be required without property lists.

## Usage

```
H5Pset_layout                    ( h5plist, layout = h5default("H5D") )
H5Pget_layout                    ( h5plist )
H5Pset_chunk                     ( h5plist, dim )
H5Pget_chunk                     ( h5plist )
H5Pset_deflate                   ( h5plist, level )
H5Pset_fill_value                ( h5plist, value )

H5Pfill_value_defined            ( h5plist )
H5Pset_fill_time             ( h5plist, fill_time = h5default("H5D_FILL_TIME") )
H5Pget_fill_time                 ( h5plist )
H5Pset_alloc_time           ( h5plist, alloc_time = h5default("H5D_ALLOC_TIME") )
H5Pget_alloc_time                ( h5plist )
```

## Arguments

| | |
|---|---|
| h5plist | An object of class H5IdComponent representing a H5 property list identifier of class H5P_DATASET_CREATE. See H5Pcreate or H5Pcopy to create an object of this kind. |
| layout | A character name of a dataset layout type. See h5const("H5D") for possible property list types. Can also be an integer representing an HDF5 dataset layout type. |
| dim | The chunk size used to store the dataset. It is an integer vector of the same length as the dataset dims. |
| level | The compression level used. An integer value between 0 (no compression) and 9 (highest and slowest compression). |
| value | Standard value for filling the dataset. The storage.mode of value has to be convertable to the dataset type by HDF5. |
| fill_time | A character name of a H5D_FILL_TIME type. See h5const("H5D_FILL_TIME") for possible values. Can also be an integer representing an HDF5 H5D_FILL_TIME type. |
| alloc_time | A character name of a H5D_ALLOC_TIME type. See h5const("H5D_ALLOC_TIME") for possible property list types. Can also be an integer representing an HDF5 H5D_ALLOC_TIME type. |

## Details

Interface to the HDF5 C-library libhdf5. See https://portal.hdfgroup.org/display/HDF5/Dataset+Creation+Properties for further details. See H5P for documentation of more functions to manupulate property lists of other property list classes.

## Value

The functions return the standard return value from their respective C-functions.

## Author(s)

Bernd Fischer

## References

https://portal.hdfgroup.org/display/HDF5

## See Also

rhdf5, H5P

## Examples

```
pid <- H5Pcreate("H5P_DATASET_CREATE")
H5Pset_fill_time( pid, "H5D_FILL_TIME_ALLOC" )
H5Pset_chunk(pid, c(1000,1,1))
H5Pset_deflate(pid, 6)

H5Pget_layout( pid )
H5Pfill_value_defined( pid )
H5Pget_fill_time( pid )
```

```
H5Pget_alloc_time( pid )
H5Pget_chunk( pid)

H5Pclose(pid)
```

---

HDF5 Dataset Interface

*HDF5 Dataset Interface*

---

## Description

These functions create and manipulate dataset objects, and set and retrieve their constant or persistent properties.

## Usage

```
H5Dcreate       (h5loc, name, dtype_id, h5space, lcpl=NULL, dcpl=NULL, dapl=NULL)
H5Dopen             (h5loc, name, dapl=NULL)
H5Dclose            (h5dataset)
H5Dget_space        (h5dataset)
H5Dget_type         (h5dataset)
H5Dget_create_plist (h5dataset)
H5Dget_storage_size (h5dataset)
H5Dread             (h5dataset, h5spaceFile = NULL, h5spaceMem = NULL,
                     buf = NULL, compoundAsDataFrame = TRUE,
                     bit64conversion, drop = FALSE)
H5Dwrite            (h5dataset, buf, h5spaceMem = NULL, h5spaceFile = NULL)
H5Dset_extent       (h5dataset, size)
```

## Arguments

| | |
|---|---|
| h5loc | An object of class H5IdComponent representing a H5 location identifier (file or group). See H5Fcreate, H5Fopen, H5Gcreate, H5Gopen to create an object of this kind. |
| name | Name of the dataset (character). |
| dtype_id | A character name of a datatype. See h5const("H5T") for possible datatypes. Can also be an integer representing an HDF5 datatype. |
| h5space | An object of class H5IdComponent representing a H5 dataspace. See H5Dget_space, H5Screate_simple, H5Screate to create an object of this kind. |
| h5dataset | An object of class H5IdComponent representing a H5 dataset. See H5Dcreate, H5Dopen to create an object of this kind. |
| h5spaceFile,h5spaceMem | An object of class H5IdComponent representing a H5 dataspace. See H5Dget_space, H5Screate_simple, H5Screate to create an object of this kind. The dimensions of the dataset in the file and in memory. The dimensions in file and in memory are interpreted in an R-like manner. The first dimension is the fastest changing dimension. When reading the file with a C-program (e.g. HDFView) the order of dimensions will invert, because in C the fastest changing dimension is the last one. |

buf
: Reading and writing buffer containing the data to written/read. When using the buffer for reading, the buffer size has to fit the size of the memory space h5spaceMem. No extra memory will be allocated for the data. A pointer to the same data is returned.

compoundAsDataFrame
: If true, a compound datatype will be coerced to a data.frame. This is not possible, if the dataset is multi-dimensional. Otherwise the compound datatype will be returned as a list. Nested compound data types will be returned as a nested list.

bit64conversion
: Defines, how 64-bit integers are converted. Internally, R does not support 64-bit integers. All integers in R are 32-bit integers. By setting bit64conversion='int', a coercing to 32-bit integers is enforced, with the risc of data loss, but with the insurance that numbers are represented as integers. bit64conversion='double' coerces the 64-bit integers to floating point numbers. doubles can represent integers with up to 54-bits, but they are not represented as integer values anymore. For larger numbers there is again a data loss. bit64conversion='bit64' is recommended way of coercing. It represents the 64-bit integers as objects of class 'integer64' as defined in the package 'bit64'. Make sure that you have installed 'bit64'. The datatype 'integer64' is not part of base R, but defined in an external package. This can produce unexpected behaviour when working with the data.

drop
: (logical) If TRUE, the HDF5 object is read as a vector with NULL dim attributes.

size
: An integer vector with the new dimension of the dataset. Calling this function is only valid for chunked datasets.

lcpl
: An object of class H5IdComponent representing a H5 link creation property list. See H5Pcreate, H5Pcopy to create an object of this kind.

dcpl
: An object of class H5IdComponent representing a H5 dataset creation property list. See H5Pcreate, H5Pcopy to create an object of this kind.

dapl
: An object of class H5IdComponent representing a H5 dataset access property list. See H5Pcreate, H5Pcopy to create an object of this kind.

## Details

Interface to the HDF5 C-library libhdf5. See https://portal.hdfgroup.org/display/HDF5/Datasets for further details.

## Value

H5Dcreate and H5Dopen return an object of class H5IdComponent represinting a H5 dataset identifier.

H5Dget_space returns an object of class H5IdComponent representing a H5 dataspace identifier.

H5Dread returns an array with the read data.

The other functions return the standard return value from their respective C-functions.

## Author(s)

Bernd Fischer

## References

https://portal.hdfgroup.org/display/HDF5

## See Also

rhdf5

## Examples

```
# write a dataset
fid <- H5Fcreate("ex_H5D.h5")
fid
sid <- H5Screate_simple(c(10,5,3))
sid
did <- H5Dcreate(fid, "A", "H5T_STD_I32LE", sid)
did
H5Dwrite(did, 1L:150L, h5spaceMem = sid, h5spaceFile = sid)
H5Dclose(did)
H5Sclose(sid)
H5Fclose(fid)

# read a dataset
fid <- H5Fopen("ex_H5D.h5")
fid
did <- H5Dopen(fid, "A")
did
sid <- H5Dget_space(did)
sid
B <- H5Dread(did)
B
H5Dclose(did)
H5Sclose(sid)
H5Fclose(fid)

# write a subarray
fid <- H5Fopen("ex_H5D.h5")
fid
did <- H5Dopen(fid, "A")
did
sid <- H5Dget_space(did)
sid
H5Sselect_index(sid, list(1:3,2:4,2))
sidmem <- H5Screate_simple(c(3,3,1))
sidmem
A = array(-801:-809,dim=c(3,3,1))
H5Dwrite(did, A, h5spaceMem = sidmem, h5spaceFile = sid)
H5Dread(did)
H5Sclose(sid)
H5Dclose(did)
H5Sclose(sidmem)
H5Fclose(fid)
```

---

HDF5 Dataspace Interface

*HDF5 Dataspace Interface*

---

## Description

These functions create and manipulate the dataspace in which to store the elements of a dataset.

## Usage

```
H5Screate                 (type = h5default("H5S"), native = FALSE)
H5Screate_simple          (dims, maxdims, native = FALSE)
H5Scopy                   (h5space)
H5Sclose                  (h5space)
H5Sis_simple              (h5space)
H5Sget_simple_extent_dims (h5space)
H5Sset_extent_simple      (h5space, dims, maxdims)
H5Sselect_hyperslab       (h5space, op = h5default("H5S_SELECT"),
                           start = NULL, stride = NULL, count = NULL,
                           block = NULL)
H5Sselect_index           (h5space, index)
H5Sunlimited              ()
```

## Arguments

| | |
|---|---|
| type | See h5const("H5S") for possible types. |
| dims | Dimension of the dataspace. This argument is similar to the dim attribute of an array. When viewing the HDF5 dataset with an C-program (e.g. HDFView), the dimensions appear in inverted order, because the fastest changing dimension in R is the first one, and in C its the last one. |
| maxdims | Maximum extension of the dimension of the dataset in the file. If not provided, it is set to dims. |
| native | An object of class logical. If TRUE, array-like objects are treated as stored in HDF5 row-major rather than R column-major orientation. Using native = TRUE increases HDF5 file portability between programming languages. A file written with native = TRUE should also be read with native = TRUE. |
| h5space | An object of class H5IdComponent representing a H5 dataspace identifier. See H5Dget_space, H5Screate_simple, H5Screate to create an object of this kind. |
| index | A list of integer indices. The length of the list corresponds to the number of dimensions of the HDF5 array. |
| op | See h5const("H5S_SELECT") for possible arguments. |
| start | The start coordinate of a hyperslab (similar to subsetting in R). Counting is R-style 1-based. |
| stride | The stride of the hypercube. Read the introduction http://ftp.hdfgroup.org/HDF5/Tutor/phypecont.html before using this argument. R behaves like Fortran in this example. |
| count | The number of blocks to be written. |
| block | The block size of the hyperslab. Read the introduction http://ftp.hdfgroup.org/HDF5/Tutor/phypecont.html before using this argument. R behaves like Fortran in this example. |

## Details

Interface to the HDF5 C-library libhdf5. See https://portal.hdfgroup.org/display/HDF5/Dataspaces for further details.

As an introduction to use hyperslabs see these tutorials: See these introductions to hyperslabs: https://support.hdfgroup.org/HDF5/Tutor/selectsimple.html, https://support.hdfgroup.org/HDF5/Tutor/select.html and http://ftp.hdfgroup.org/HDF5/Tutor/phypecont.html.

Please note that in R the first dimension is the fastest changing dimension. When viewing the HDF5 datasets with any C-program (e.g. HDFView), the order of dimensions is inverted. In the R interface counting starts with 1, whereas in the C-programs (e.g. HDFView) counting starts with 0.

`H5Sselect_index` is not part of the standard HDF5 C interface. It performes an iterative call to H5select_points by iterating through the given index positions. This function avoids a for loop in R. If a list element is NULL, all elements of the respective dimension are considered.

## Value

`H5Screate`, `H5Screate_simple`, and `H5Scopy` return an object of class `H5IdComponent` representing a dataspace.

`H5Sis_simple` returns a boolean.

`H5Sget_simple_extent_dims` returns an integer vector.

`H5Sunlimited` is a simple macro to return the constant `H5S_UNLIMITED` that can be provided to the `maxdims` arguments of `H5Screate_simple` to create and extensible dataspace.

The other functions return the standard return value from their respective C-functions.

## Author(s)

Bernd Fischer

## References

https://portal.hdfgroup.org/display/HDF5

## See Also

rhdf5

## Examples

```
sid <- H5Screate_simple(c(10,5,3))
sid
H5Sis_simple(sid)
H5Sget_simple_extent_dims(sid)

# select a subarray (called hyperslab in the hdf5 community).
# The next h5write can use this to write a subarray
H5Sselect_index(sid, list(1:3,2:4,2))

# always close dataspaces after usage to free recources
H5Sclose(sid)
sid
```

HDF5 Datatype Interface

*HDF5 Datatype Interface*

## Description

These functions create and manipulate the datatype which describes elements of a dataset.

## Usage

```
H5Tcopy      (dtype_id = h5default(type = "H5T"))
H5Tset_size  (dtype_id = h5default(type = "H5T"), size)
H5Tget_size  (dtype_id)
H5Tset_strpad(dtype_id, strpad = "NULLPAD")
H5Tget_strpad(dtype_id)
```

## Arguments

| | |
|---|---|
| dtype_id | A character name of a datatype. See `h5const("H5T")` for possible datatypes. Can also be an integer representing an HDF5 datatype. |
| size | The total size in bytes. |
| strpad | Character string repsenting the type of padding to be used. Valid options are 'NULLTERM', 'NULLPAD', 'SPACEPAD' |

## Details

Interface to the HDF5 C-library libhdf5. See https://portal.hdfgroup.org/display/HDF5/Datatypes for further details.

## Value

The functions return the standard return value from their respective C-functions.

## Author(s)

Bernd Fischer

## References

https://portal.hdfgroup.org/display/HDF5

## See Also

rhdf5

## Examples

```
# create character datatype with string length 10
tid <- H5Tcopy("H5T_C_S1")
H5Tset_size(tid, 10L)

# List all predefined types implemented in the R-interface
h5const("H5T")

# List all available type classes (not all of them are implemented)
h5const("H5T_CLASS")
```

---

# HDF5 File Interface *HDF5 File Interface*

---

## Description

These functions are designed to provide file-level access to HDF5 files.

## Usage

```
H5Fcreate           (name, flags = h5default("H5F_ACC"), fcpl = NULL,
                     fapl = NULL, native = FALSE)
H5Fopen             (name, flags = h5default("H5F_ACC_RD"), native = FALSE)
H5Fclose            (h5file)
H5Fflush            (h5file, scope = h5default("H5F_SCOPE"))
H5Fis_hdf5          (name, showWarnings = TRUE)
H5Fget_filesize     (h5file)
H5Fget_name         (h5obj)
H5Fget_create_plist (h5file)
H5Fget_access_plist (h5file)
```

## Arguments

| | |
|---|---|
| name | The filename of the HDF5 file. |
| flags | See `h5const("H5F_ACC")` for possible arguments. |
| fcpl | An object of class `H5IdComponent` representing a H5 file creation property list. See `H5Pcreate`, `H5Pcopy` to create an object of this kind. |
| fapl | An object of class `H5IdComponent` representing a H5 file access property list. See `H5Pcreate`, `H5Pcopy` to create an object of this kind. |
| native | An object of class `logical`. If TRUE, array-like objects are treated as stored in HDF5 row-major rather than R column-major orientation. Using `native = TRUE` increases HDF5 file portability between programming languages. A file written with `native = TRUE` should also be read with `native = TRUE`. |
| h5file | An object of class `H5IdComponent` representing a H5 file identifier as created by `H5Fcreate` or `H5Fopen`. |
| scope | See `h5const("H5F_ACC_RD")` for possible arguments. |
| showWarnings | If TRUE (default), a warning is given if an open HDF5 handle exists for this file. |
| h5obj | An object of class `H5IdComponent` representing a H5 object identifier (file, group, or dataset). See `H5Fcreate`, `H5Fopen`, `H5Gcreate`, `H5Gopen`, `H5Dcreate`, or `H5Dopen` to create an object of this kind. |

### Details

Interface to the HDF5 C-library libhdf5. See https://portal.hdfgroup.org/display/HDF5/Files for further details.

### Value

H5Fcreate and H5Fopen return an object of class H5IdComponent representing a H5 file identifier.

H5Fis_hdf5 returns TRUE, if the file is an HDF5 file, or FALSE otherwise. In the case the file doesn't exist, NA is returned.

The other functions return the standard return value from their respective C-functions.

### Author(s)

Bernd Fischer

### References

https://portal.hdfgroup.org/display/HDF5

### See Also

rhdf5

### Examples

```
fid <- H5Fcreate("ex_H5F.h5")
fid
H5Fclose(fid)
fid2 <- H5Fopen("ex_H5F.h5")
fid2
H5Fclose(fid2)
```

---

## HDF5 General Library Functions

*HDF5 General Library Functions*

---

### Description

These low level functions provide general general libary functions for HDF5.

### Usage

```
H5open             ()
H5close            ()
H5garbage_collect  ()
H5get_libversion   ()
```

### Details

These low level functions provide general general libary functions for HDF5 C-library libhdf5. See https://portal.hdfgroup.org/display/HDF5/Library for further details.

### Value

`H5open` initializes the HDF5 library. `H5close` flushes all data to disk, closes all open identifiers, and cleans up memory. `H5garbage_collect` cleans up memory. `H5get_libversion` returns the version number of the HDF5 C-library.

### Author(s)

Bernd Fischer

### References

https://portal.hdfgroup.org/display/HDF5

### See Also

`h5version`, rhdf5

### Examples

```
# test for file locking
if( !h5testFileLocking( tempdir() ) ) {
      h5disableFileLocking()
}

H5open()
H5close()
H5garbage_collect()
H5get_libversion()
```

---

## `HDF5 Group Interface` *HDF5 Group Interface*

---

### Description

The Group interface functions create and manipulate groups of objects in an HDF5 file.

### Usage

```
H5Gcreate      (h5loc, name)
H5Gcreate_anon (h5loc)
H5Gopen        (h5loc, name)
H5Gclose       (h5group)
H5Gget_info    (h5loc)
H5Gget_info_by_idx  (h5loc, n, group_name = ".",
                        index_type = h5default("H5_INDEX"),
       order = h5default("H5_ITER"))
H5Gget_info_by_name (h5loc, group_name)
```

## Arguments

| | |
|---|---|
| h5loc | An object of class H5IdComponent representing a H5 location identifier (file or group). See H5Fcreate, H5Fopen, H5Gcreate, H5Gopen to create an object of this kind. |
| name | Name of the group. |
| h5group | An object of class H5IdComponent representing a H5 group identifier. See H5Gcreate, H5Gopen to create an object of this kind. |
| n | Position in the index of the group for which information is retrieved (Counting is 1-based). |
| group_name | An additional group name specifying the group for which information is sought. It is interpreted relative to h5loc. |
| index_type | See h5const("H5_INDEX") for possible arguments. |
| order | See h5const("H5_ITER") for possible arguments. |

## Details

Interface to the HDF5 C-library libhdf5. See https://portal.hdfgroup.org/display/HDF5/Groups for further details.

## Value

H5Gcreate, H5Gcreate_anon, and H5Gopen return an object of class H5IdComponent representing a H5 group identifier.

H5Gget_info, H5Gget_info_by_idx, and H5Gget_info_by_name return a list with the group information.

The other functions return the standard return value from their respective C-functions.

## Author(s)

Bernd Fischer

## References

https://portal.hdfgroup.org/display/HDF5

## See Also

rhdf5

## Examples

```
fid <- H5Fcreate("ex_H5G.h5")
gid <- H5Gcreate(fid, "foo")
gid
H5Gget_info(gid)
H5Gclose(gid)

H5Gget_info_by_idx(fid,1)
H5Gget_info_by_name(fid,"foo")

H5Fclose(fid)
```

`HDF5 Identifier Interface`
*HDF5 Identifier Interface*

## Description

These functions provides tools for working with object identifiers and object names.

## Usage

```
H5Iget_type(h5identifier)
H5Iget_name(h5obj)
H5Iis_valid(h5identifier)
```

## Arguments

| | |
|---|---|
| `h5identifier` | An object of class `H5IdComponent` representing a H5 identifier (file, group, dataset, dataspace, datatype, attribute). See e.g. `H5Fcreate`, `H5Fopen`, `H5Gcreate`, `H5Gopen`, `H5Dcreate`, `H5Dopen` to create an object of this kind. |
| `h5obj` | An object of class `H5IdComponent` representing a H5 object identifier (file, group, or dataset). See `H5Fcreate`, `H5Fopen`, `H5Gcreate`, `H5Gopen`, `H5Dcreate`, or `H5Dopen` to create an object of this kind. |

## Details

Interface to the HDF5 C-library libhdf5. See `https://portal.hdfgroup.org/display/HDF5/Identifiers` for further details.

## Value

`H5Iget_type` returns the type of the H5 identifier, `H5Iget_name` the name of the object, and `H5Iis_valid` checks if the object is a valid H5 identifier.

## Author(s)

Bernd Fischer

## References

`https://portal.hdfgroup.org/display/HDF5`

## See Also

rhdf5

## Examples

```
# create an hdf5 file and write something
h5createFile("ex_H5I.h5")
h5createGroup("ex_H5I.h5","foo")
B = array(seq(0.1,2.0,by=0.1),dim=c(5,2,2))
h5write(B, "ex_H5I.h5","foo/B")

# reopen file and dataset and get object info
fid <- H5Fopen("ex_H5I.h5")
oid = H5Oopen(fid, "foo")
H5Iget_type(oid)
H5Oclose(oid)
H5Fclose(fid)
```

---

# HDF5 Link Create Property List Interface

*HDF5 Link Create Property List Interface*

---

## Description

The functions, macros, and subroutines listed here are used to manipulate link creation property list objects in various ways, including to reset property values. With the use of property lists, HDF5 functions have been implemented and can be used in applications with many fewer parameters than would be required without property lists.

## Usage

```
H5Pset_char_encoding              ( h5plist, encoding  = h5default("H5T_CSET"))
H5Pget_char_encoding               ( h5plist )
H5Pset_create_intermediate_group   ( h5plist, crt_intermed_group )
H5Pget_create_intermediate_group   ( h5plist )
```

## Arguments

| | |
|---|---|
| h5plist | An object of class H5IdComponent representing a H5 property list identifier of class H5P_LINK_CREATE. See H5Pcreate or H5Pcopy to create an object of this kind. |
| encoding | A character name of an encoding type. See h5const("H5T_CSET") for possible property list types. Can also be an integer representing an HDF5 encoding type. |
| crt_intermed_group | Logical. If TRUE intermediate groups are created. |

## Details

Interface to the HDF5 C-library libhdf5. See https://portal.hdfgroup.org/display/HDF5/Attribute+and+Link+Creation+Properties for further details. See H5P for documentation of more functions to manupulate property lists of other property list classes.

## Value

The functions return the standard return value from their respective C-functions.

### Author(s)

Bernd Fischer

### References

https://portal.hdfgroup.org/display/HDF5

### See Also

rhdf5, H5P

### Examples

```
pid <- H5Pcreate("H5P_LINK_CREATE")
H5Pset_char_encoding( pid, "H5T_CSET_ASCII")
H5Pget_char_encoding( pid)
H5Pset_create_intermediate_group(pid, TRUE)
H5Pget_create_intermediate_group(pid)
H5Pclose(pid)
```

---

## HDF5 Link Interface *HDF5 Link Interface*

---

### Description

The Link interface, H5L, functions create and manipulate links in an HDF5 group. This interface includes functions that enable the creation and use of user-defined link classes.

### Usage

```
H5Lcreate_external (target_file_name, target_obj_name, link_loc, link_name)
H5Lexists          (h5loc, name)
H5Lget_info        (h5loc, name)
H5Ldelete          (h5loc, name)
H5Lcopy           (h5loc, name, h5loc_dest, name_dest, lcpl = NULL, lapl = NULL )
H5Lmove           (h5loc, name, h5loc_dest, name_dest, lcpl = NULL, lapl = NULL )
```

### Arguments

| | |
|---|---|
| target_file_name | the relative or absoulte target file name containing the target object. |
| target_obj_name | the absolute path and name of the target object within the target file. |
| link_loc | An object of class H5IdComponent representing a H5 location identifier (file or group) where the new link is placed. |
| link_name | The name of the new link. |
| h5loc | An object of class H5IdComponent representing a H5 location identifier (file or group). See H5Fcreate, H5Fopen, H5Gcreate, H5Gopen to create an object of this kind. |
| h5loc_dest | An object of class H5IdComponent representing the destination file or group where a copied or moved link should be created. |

| | |
|---|---|
| name | The name of the link to be checked, copied or moved. |
| name_dest | The name of the link to be created when copying or moving. |
| lcpl | Link creation property list. |
| lapl | Link access property list. |

## Details

Interface to the HDF5 C-library libhdf5. See https://portal.hdfgroup.org/display/HDF5/Links for further details.

If name consists of a relative path containing group names, the function H5Lexists checks recursively if the links exists which is a different behaviour to the C-function.

## Value

H5Lexists returns boolean TRUE if the link exists and FALSE otherwise.

H5Lget_info returns a list with the entries of the C-structure H5L_info_t.

## Author(s)

Bernd Fischer, Mike Smith

## References

https://portal.hdfgroup.org/display/HDF5

## See Also

rhdf5

## Examples

```
# create an hdf5 file and a group
h5createFile("ex_H5L.h5")
h5createGroup("ex_H5L.h5","foo")

# reopen file and get link info
fid <- H5Fopen("ex_H5L.h5")
H5Lexists(fid, "foo")
H5Lexists(fid, "baa")
H5Lget_info(fid, "foo")

H5Ldelete(fid, "foo")
H5Lexists(fid, "foo")

H5Fclose(fid)
```

`HDF5 Object Interface` *HDF5 Object Interface*

## Description

The Object interface, H5O, functions manipulate objects in an HDF5 file. This interface is designed to be used in conjunction with the Links interface (H5L).

## Usage

```
H5Oopen  (h5loc, name)
H5Oclose (h5obj)
H5Oget_num_attrs(h5obj)
H5Oget_num_attrs_by_name(h5loc, name)
```

## Arguments

| | |
|---|---|
| `h5obj` | An object of class `H5IdComponent` representing a H5 object identifier (file, group, or dataset). See `H5Fcreate`, `H5Fopen`, `H5Gcreate`, `H5Gopen`, `H5Dcreate`, or `H5Dopen` to create an object of this kind. |
| `h5loc` | An object of class `H5IdComponent` representing a H5 location identifier (file or group). See `H5Fcreate`, `H5Fopen`, `H5Gcreate`, `H5Gopen` to create an object of this kind. |
| `name` | The name of the link to be checked. |

## Details

Interface to the HDF5 C-library libhdf5. See `https://portal.hdfgroup.org/display/HDF5/Objects` for further details.

## Value

`H5Oopen` opens an object (a file, group, or dataset) and returns an object of class `H5IdComponent`. `H5Oclose` closed the object again. `H5Oget_num_attrs` and `H5Oget_num_attrs_by_name` return the number of attributes of an object.

## Author(s)

Bernd Fischer

## References

`https://portal.hdfgroup.org/display/HDF5`

## See Also

rhdf5

## Examples

```
# create an hdf5 file and write something
h5createFile("ex_H5O.h5")
h5createGroup("ex_H5O.h5","foo")
B = array(seq(0.1,2.0,by=0.1),dim=c(5,2,2))
h5write(B, "ex_H5O.h5","foo/B")

# reopen file and dataset and get object info
fid <- H5Fopen("ex_H5O.h5")
oid = H5Oopen(fid, "foo")
H5Oget_num_attrs(oid)
H5Oclose(oid)
H5Fclose(fid)
```

---

HDF5 Property List Interface

*HDF5 Property List Interface*

---

## Description

The functions, macros, and subroutines listed here are used to manipulate property list objects in various ways, including to reset property values. With the use of property lists, HDF5 functions have been implemented and can be used in applications with many fewer parameters than would be required without property lists.

## Usage

```
H5Pcreate             (type = h5default("H5P"), native = FALSE)
H5Pcopy               (h5plist)
H5Pget_class          (h5plist)
H5Pclose              (h5plist)
H5Pclose_class        (h5plistclass)
H5Pequal              (h5plistclass1, h5plistclass2)
H5Pset_libver_bounds  (h5plist,
                       libver_low = "H5F_LIBVER_EARLIEST",
                       libver_high = "H5F_LIBVER_LATEST")
H5Pget_libver_bounds  (h5plist)
H5Pset_obj_track_times(h5plist, track_times = TRUE)
H5Pget_obj_track_times(h5plist)
```

## Arguments

| | |
|---|---|
| type | A character name of a property list type. See `h5const("H5P")` for possible property list types. Can also be an integer representing an HDF5 property list type. |
| native | An object of class `logical`. If TRUE, array-like objects are treated as stored in HDF5 row-major rather than R column-major orientation. Using `native = TRUE` increases HDF5 file portability between programming languages. A file written with `native = TRUE` should also be read with `native = TRUE`. |

h5plist — An object of class H5IdComponent representing a H5 property list identifier. See H5Pcreate or H5Pcopy to create an object of this kind.

h5plistclass, h5plistclass1, h5plistclass2 — An object of class H5IdComponent representing a H5 property list class identifier. See H5Pget_class to create an object of this kind.

libver_low — A character value representing the lower bound on the library version for reading or writing the HDF5 file. See h5const("H5F_LIBVER") for possible arguments. Default is "H5F_LIBVER_EARLIEST"

libver_high — A character value representing the higher bound on the library version for reading or writing the HDF5 file. See h5const("H5F_LIBVER") for possible arguments. Default is "H5F_LIBVER_LATEST"

track_times — logical specifying whether times associated with an object are being recorded.

## Details

Interface to the HDF5 C-library libhdf5. See https://portal.hdfgroup.org/display/HDF5/Property+Lists for further details.

## Value

H5Pcreate and H5Pcopy return an object of class H5IdComponent representing a H5 property list identifier.

H5Pget_class returns an object of class H5IdComponent representing a H5 property list class identifier.

The other functions return the standard return value from their respective C-functions.

## Author(s)

Bernd Fischer, Mike L. Smith

## References

https://portal.hdfgroup.org/display/HDF5

## See Also

rhdf5, H5P_DATASET_CREATE

## Examples

```
pid <- H5Pcreate()
pid2 <- H5Pcopy(pid)
pclid <- H5Pget_class(pid)
H5Pclose_class(pclid)
H5Pclose(pid)
H5Pclose(pid2)
```

rhdf5 *Package overview*

## Description

rhdf5 is an interface to the HDF5 library. The R-package contains the complete HDF5 library, thus no further installation of external packages is necessary.

There are a number of high level R functions that provide a convinient way of accessing HDF5 file as well as R interfaces to a number of functions in the C-library.

## Package content

HDF5 file, group, dataset creation

- h5createFile
- h5createGroup
- h5createDataset

HDF5 file content listing

- h5ls
- h5dump

Reading and writing data

- h5read, h5write
- h5dump, h5save

HDF5 constants

- h5const, h5default, h5constType

HDF5 version number

- h5version

Low level interface to HDF5 C-library (for expert users only!):

- general HDF5 functions (H5open / H5close / H5garbage_collect / H5get_libversion)
- HDF5 File Interface (H5Fcreate, H5Fopen / H5Fclose / H5Fflush)
- HDF5 Group Interface (H5Gcreate, H5Gcreate_anon, H5Gopen / H5Gclose / H5Gget_info, H5Gget_info_by_idx, H5Gget_info_by_name)
- HDF5 Link Interface (H5Lexists, H5Lget_info)
- HDF5 Object Interface (H5Oopen, H5Oclose, H5Oget_num_attrs, H5Oget_num_attrs_by_name)
- HDF5 Identifier Interface (H5Iget_type, H5Iget_name, H5Iis_valid)
- HDF5 Dataset Interface (H5Dcreate, H5Dopen / H5Dclose / H5Dget_space / H5Dread, H5Dwrite)
- HDF5 Attribute Interface (H5Acreate, H5Aopen, H5Aopen_by_idx, H5Aopen_by_name / H5Aclose, H5Adelete / H5Aexists / H5Aget_name, H5Aget_space, H5Aget_type / H5Aread, H5Awrite)
- HDF5 Dataspace Interface (H5Screate, H5Screate_simple, H5Scopy / H5Sclose / H5Sis_simple, H5Sget_simple_extent_dims / H5Sselect_hyperslab)
- HDF5 Datatype Interface (H5Tcopy, H5Tset_size)

## Authors

R-interface by

Bernd Fischer, <bernd.fischer@embl.de> EMBL - European Molecular Biology Laboratory Heidelberg Germany

The package contains the HDF5 library (https://portal.hdfgroup.org/display/HDF5).

## Examples

```
h5createFile("ex_hdf5file.h5")

# create groups
h5createGroup("ex_hdf5file.h5","foo")
h5createGroup("ex_hdf5file.h5","foo/foobaa")

# write a matrix
B = array(seq(0.1,2.0,by=0.1),dim=c(5,2,2))
attr(B, "scale") <- "liter"
h5write(B, "ex_hdf5file.h5","foo/B")

# read a matrix
E = h5read("ex_hdf5file.h5","foo/B")

# list content of hdf5 file
h5ls("ex_hdf5file.h5")

# write and read submatrix
h5createDataset("ex_hdf5file.h5", "foo/S", c(5,8), storage.mode = "integer", chunk=c(5,1), level=7)
h5write(matrix(1:5,nr=5,nc=1), file="ex_hdf5file.h5", name="foo/S", index=list(NULL,1))
h5read("ex_hdf5file.h5", "foo/S")
h5read("ex_hdf5file.h5", "foo/S", index=list(2:3,2:3))
```

---

# [-methods *Methods for Function* [ *in Package* **rhdf5**

---

## Description

Methods for function [ in package **rhdf5**.

## Methods

signature(x = "H5IdComponent") Subsetting of an HDF5 dataset. The function reads a subset of an HDF5 dataset. The given dimensions have to fit the dimensions of the HDF5 dataset.

## [<--methods *Methods for Function* [<- *in Package* **rhdf5**

### Description

Methods for function [<- in package **rhdf5**.

### Methods

signature(x = "H5IdComponent") Subsetting of an HDF5 dataset. The function writes an R data object to a subset of an HDF5 dataset. The given dimensions have to fit the dimensions of the HDF5 dataset. The HDF5 dataset has to be created beforehand, e.g. by h5createDataset().

## $-methods *Methods for Function* $ *in Package* **rhdf5**

### Description

Methods for function $ in package **rhdf5**.

### Methods

signature(x = "H5IdComponent") Reads the HDF5 object e2 in the HDF5 location e1. e1 can either be a file handle as returned by H5Fopen() or a group handle as e.g. returned by h5f$g1 or h5f$'/g1/g2's.

## $<--methods ~~ *Methods for Function* $<- *in Package* **rhdf5** ~~

### Description

Methods for function $<- in package **rhdf5**.

### Methods

signature(x = "H5IdComponent") Writes the assigned object to to the HDF5 file at location e1. e1 can either be a file handle as returned by H5Fopen() or a group handle as e.g. returned by h5f$g1 or h5f$'/g1/g2's. The storage.mode of the assigned object has to be compatible to the datatype of the HDF5 dataset. The dimension of the assigned object have to be identical the dimensions of the HDF5 dataset. To create a new HDF5 dataset with specific properties (e.g. compression level or chunk size), please use the function h5createDataset first.

`&-methods` *Methods for Function* & *in Package* **rhdf5**

## Description

Methods for function & in package **rhdf5**.

## Methods

`signature(e1 = "H5IdComponent", e2 = "ANY")` Returns a group handle or dataset handle for the group or dataset e2 in the HDF5 location e1. e1 can either be a file handle as returned by H5Fopen() or a group handle as e.g. returned by h5f$g1 or h5f$'/g1/g2's.

# Index


∗ **IO**
[-methods, 45
[<--methods, 46
$-methods, 46
$<--methods, 46
&-methods, 47
h5closeAll, 2
h5const, 3
h5createAttribute, 4
h5createDataset, 5
h5createFile, 7
h5createGroup, 8
h5errorHandling, 10
h5listIdentifier, 12
h5ls, 13
h5save, 14
h5set_extent, 15
h5testFileLocking, 16
h5version, 18
h5write, 18
HDF5 Attribute Interface, 22
HDF5 Dataset Access Property List Interface, 24
HDF5 Dataset Create Property List Interface, 25
HDF5 Dataset Interface, 27
HDF5 Dataspace Interface, 29
HDF5 Datatype Interface, 32
HDF5 File Interface, 33
HDF5 General Library Functions, 34
HDF5 Group Interface, 35
HDF5 Identifier Interface, 37
HDF5 Link Create Property List Interface, 38
HDF5 Link Interface, 39
HDF5 Object Interface, 41
HDF5 Property List Interface, 42
∗ **classes**
H5IdComponent-class, 11
∗ **file**
[-methods, 45
[<--methods, 46
$-methods, 46
$<--methods, 46
&-methods, 47
h5closeAll, 2
h5const, 3
h5createAttribute, 4
h5createDataset, 5
h5createFile, 7
h5createGroup, 8
h5delete, 9
h5errorHandling, 10
h5listIdentifier, 12
h5ls, 13
h5save, 14
h5set_extent, 15
h5testFileLocking, 16
h5version, 18
h5write, 18
HDF5 Attribute Interface, 22
HDF5 Dataset Access Property List Interface, 24
HDF5 Dataset Create Property List Interface, 25
HDF5 Dataset Interface, 27
HDF5 Dataspace Interface, 29
HDF5 Datatype Interface, 32
HDF5 File Interface, 33
HDF5 General Library Functions, 34
HDF5 Group Interface, 35
HDF5 Identifier Interface, 37
HDF5 Link Create Property List Interface, 38
HDF5 Link Interface, 39
HDF5 Object Interface, 41
HDF5 Property List Interface, 42
∗ **interface**
[-methods, 45
[<--methods, 46
$-methods, 46
$<--methods, 46
&-methods, 47
h5const, 3
h5createAttribute, 4
h5createDataset, 5

h5createFile, 7
h5createGroup, 8
h5errorHandling, 10
h5listIdentifier, 12
h5ls, 13
h5save, 14
h5set_extent, 15
h5version, 18
h5write, 18
HDF5 Attribute Interface, 22
HDF5 Dataset Access Property List Interface, 24
HDF5 Dataset Create Property List Interface, 25
HDF5 Dataset Interface, 27
HDF5 Dataspace Interface, 29
HDF5 Datatype Interface, 32
HDF5 File Interface, 33
HDF5 General Library Functions, 34
HDF5 Group Interface, 35
HDF5 Identifier Interface, 37
HDF5 Link Create Property List Interface, 38
HDF5 Link Interface, 39
HDF5 Object Interface, 41
HDF5 Property List Interface, 42
∗ **methods**
[-methods, 45
[<--methods, 46
$-methods, 46
$<--methods, 46
&-methods, 47
∗ **package**
rhdf5, 44
∗ **programming**
[-methods, 45
[<--methods, 46
$-methods, 46
$<--methods, 46
&-methods, 47
h5const, 3
h5createAttribute, 4
h5createDataset, 5
h5createFile, 7
h5createGroup, 8
h5errorHandling, 10
h5listIdentifier, 12
h5ls, 13
h5save, 14
h5set_extent, 15
h5version, 18
h5write, 18
HDF5 Attribute Interface, 22
HDF5 Dataset Access Property List Interface, 24
HDF5 Dataset Create Property List Interface, 25
HDF5 Dataset Interface, 27
HDF5 Dataspace Interface, 29
HDF5 Datatype Interface, 32
HDF5 File Interface, 33
HDF5 General Library Functions, 34
HDF5 Group Interface, 35
HDF5 Identifier Interface, 37
HDF5 Link Create Property List Interface, 38
HDF5 Link Interface, 39
HDF5 Object Interface, 41
HDF5 Property List Interface, 42
[,H5IdComponent-method *([-methods)*, 45
[-methods, 45
[<--methods, 46
[<-,H5IdComponent-method *([<--methods)*, 46
$,H5IdComponent-method *($-methods)*, 46
$-methods, 46
$<--methods, 46
$<-,H5IdComponent-method *($<--methods)*, 46
&,H5IdComponent,ANY-method *(&-methods)*, 47
&-methods, 47

H5 *(HDF5 General Library Functions)*, 34
H5A *(HDF5 Attribute Interface)*, 22
H5Aclose, *44*
H5Aclose *(HDF5 Attribute Interface)*, 22
H5Acreate, *11*, *44*
H5Acreate *(HDF5 Attribute Interface)*, 22
H5Adelete, *44*
H5Adelete *(HDF5 Attribute Interface)*, 22
H5Aexists, *44*
H5Aexists *(HDF5 Attribute Interface)*, 22
H5Aget_name, *44*
H5Aget_name *(HDF5 Attribute Interface)*, 22
H5Aget_space, *44*
H5Aget_space *(HDF5 Attribute Interface)*, 22
H5Aget_type, *44*
H5Aget_type *(HDF5 Attribute Interface)*, 22
H5Aopen, *11*, *44*
H5Aopen *(HDF5 Attribute Interface)*, 22
H5Aopen_by_idx, *44*

H5Aopen_by_idx (HDF5 Attribute Interface), 22
H5Aopen_by_name, 44
H5Aopen_by_name (HDF5 Attribute Interface), 22
H5Aread, 44
H5Aread (HDF5 Attribute Interface), 22
H5Awrite, 44
H5Awrite (HDF5 Attribute Interface), 22
H5close, 2, 44
H5close (HDF5 General Library Functions), 34
h5closeAll, 2
h5const, 3, 44
h5constType, 44
h5constType (h5const), 3
h5createAttribute, 4
h5createDataset, 5, 5, 8, 9, 20, 21, 44
h5createFile, 5, 7, 7, 9, 21, 44
h5createGroup, 5, 7, 8, 8, 44
H5D (HDF5 Dataset Interface), 27
H5Dclose, 44
H5Dclose (HDF5 Dataset Interface), 27
H5Dcreate, 4, 11, 16, 19, 22, 27, 33, 37, 41, 44
H5Dcreate (HDF5 Dataset Interface), 27
h5default, 44
h5default (h5const), 3
h5delete, 9
H5Dget_create_plist (HDF5 Dataset Interface), 27
H5Dget_space, 11, 22, 27, 30, 44
H5Dget_space (HDF5 Dataset Interface), 27
H5Dget_storage_size (HDF5 Dataset Interface), 27
H5Dget_type (HDF5 Dataset Interface), 27
h5disableFileLocking (h5testFileLocking), 16
H5Dopen, 4, 11, 16, 19, 22, 27, 33, 37, 41, 44
H5Dopen (HDF5 Dataset Interface), 27
H5Dread, 20, 44
H5Dread (HDF5 Dataset Interface), 27
H5Dset_extent (HDF5 Dataset Interface), 27
h5dump, 15, 44
h5dump (h5ls), 13
H5Dwrite, 44
H5Dwrite (HDF5 Dataset Interface), 27
h5enableFileLocking (h5testFileLocking), 16
h5errorHandling, 10
H5F (HDF5 File Interface), 33
H5Fclose, 44
H5Fclose (HDF5 File Interface), 33
H5Fcreate, 4, 6, 8, 11, 13, 15–17, 19, 22, 27, 33, 36, 37, 39, 41, 44
H5Fcreate (HDF5 File Interface), 33
H5Fflush, 44
H5Fflush (HDF5 File Interface), 33
H5Fget_access_plist (HDF5 File Interface), 33
H5Fget_create_plist (HDF5 File Interface), 33
H5Fget_filesize (HDF5 File Interface), 33
H5Fget_name (HDF5 File Interface), 33
H5Fis_hdf5 (HDF5 File Interface), 33
H5Fopen, 4, 6, 8, 11, 13, 15, 16, 19, 22, 27, 33, 36, 37, 39, 41, 44
H5Fopen (HDF5 File Interface), 33
H5G (HDF5 Group Interface), 35
H5garbage_collect, 44
H5garbage_collect (HDF5 General Library Functions), 34
H5Gclose, 44
H5Gclose (HDF5 Group Interface), 35
H5Gcreate, 4, 6, 8, 11, 13, 15, 16, 19, 22, 27, 33, 36, 37, 39, 41, 44
H5Gcreate (HDF5 Group Interface), 35
H5Gcreate_anon, 44
H5Gcreate_anon (HDF5 Group Interface), 35
H5get_libversion, 44
H5get_libversion (HDF5 General Library Functions), 34
H5Gget_info, 44
H5Gget_info (HDF5 Group Interface), 35
H5Gget_info_by_idx, 44
H5Gget_info_by_idx (HDF5 Group Interface), 35
H5Gget_info_by_name, 44
H5Gget_info_by_name (HDF5 Group Interface), 35
H5Gopen, 4, 6, 8, 11, 13, 15, 16, 19, 22, 27, 33, 36, 37, 39, 41, 44
H5Gopen (HDF5 Group Interface), 35
H5I (HDF5 Identifier Interface), 37
H5IdComponent, 4, 6, 8, 12, 13, 15, 16, 19, 22–24, 26–28, 30, 31, 33, 34, 36–39, 41, 43
H5IdComponent (H5IdComponent-class), 11
H5IdComponent-class, 11
H5Iget_name, 44
H5Iget_name (HDF5 Identifier

Interface), 37
H5Iget_type, *44*
H5Iget_type (HDF5 Identifier Interface), 37
H5Iis_valid, *44*
H5Iis_valid (HDF5 Identifier Interface), 37
H5L (HDF5 Link Interface), 39
H5Lcopy (HDF5 Link Interface), 39
H5Lcreate_external (HDF5 Link Interface), 39
H5Ldelete (HDF5 Link Interface), 39
H5Lexists, *44*
H5Lexists (HDF5 Link Interface), 39
H5Lget_info, *44*
H5Lget_info (HDF5 Link Interface), 39
h5listIdentifier, 12
H5Lmove (HDF5 Link Interface), 39
h5ls, 13, *15*, *21*, *44*
H5O (HDF5 Object Interface), 41
H5Oclose, *44*
H5Oclose (HDF5 Object Interface), 41
H5Oget_num_attrs, *44*
H5Oget_num_attrs (HDF5 Object Interface), 41
H5Oget_num_attrs_by_name, *44*
H5Oget_num_attrs_by_name (HDF5 Object Interface), 41
H5Oopen, *44*
H5Oopen (HDF5 Object Interface), 41
H5open, *44*
H5open (HDF5 General Library Functions), 34
H5P, *24*, *26*, *38, 39*
H5P (HDF5 Property List Interface), 42
H5P_DATASET_ACCESS (HDF5 Dataset Access Property List Interface), 24
H5P_DATASET_CREATE, *43*
H5P_DATASET_CREATE (HDF5 Dataset Create Property List Interface), 25
H5P_LINK_CREATE (HDF5 Link Create Property List Interface), 38
H5Pall_filters_avail (HDF5 Dataset Create Property List Interface), 25
H5Pclose (HDF5 Property List Interface), 42
H5Pclose_class (HDF5 Property List Interface), 42
H5Pcopy, *24*, *26*, *28*, *33*, *38*, *43*
H5Pcopy (HDF5 Property List Interface), 42
H5Pcreate, *24*, *26*, *28*, *33*, *38*, *43*
H5Pcreate (HDF5 Property List Interface), 42
H5Pequal (HDF5 Property List Interface), 42
H5Pfill_value_defined (HDF5 Dataset Create Property List Interface), 25
H5Pget_alloc_time (HDF5 Dataset Create Property List Interface), 25
H5Pget_char_encoding (HDF5 Link Create Property List Interface), 38
H5Pget_chunk (HDF5 Dataset Create Property List Interface), 25
H5Pget_chunk_cache (HDF5 Dataset Access Property List Interface), 24
H5Pget_class, *43*
H5Pget_class (HDF5 Property List Interface), 42
H5Pget_create_intermediate_group (HDF5 Link Create Property List Interface), 38
H5Pget_external (HDF5 Dataset Create Property List Interface), 25
H5Pget_external_count (HDF5 Dataset Create Property List Interface), 25
H5Pget_fill_time (HDF5 Dataset Create Property List Interface), 25
H5Pget_fill_value (HDF5 Dataset Create Property List Interface), 25
H5Pget_filter1 (HDF5 Dataset Create Property List Interface), 25
H5Pget_filter2 (HDF5 Dataset Create Property List Interface), 25
H5Pget_filter_by_id1 (HDF5 Dataset Create Property List Interface), 25
H5Pget_filter_by_id2 (HDF5 Dataset Create Property List Interface), 25
H5Pget_layout (HDF5 Dataset Create Property List Interface), 25
H5Pget_libver_bounds (HDF5 Property List Interface), 42
H5Pget_nfilters (HDF5 Dataset Create Property List Interface), 25
H5Pget_obj_track_times (HDF5 Property List Interface), 42

H5Pmodify_filter (HDF5 Dataset Create Property List Interface), 25
H5Premove_filter (HDF5 Dataset Create Property List Interface), 25
H5Pset_alloc_time (HDF5 Dataset Create Property List Interface), 25
H5Pset_char_encoding (HDF5 Link Create Property List Interface), 38
H5Pset_chunk (HDF5 Dataset Create Property List Interface), 25
H5Pset_chunk_cache (HDF5 Dataset Access Property List Interface), 24
H5Pset_create_intermediate_group (HDF5 Link Create Property List Interface), 38
H5Pset_deflate (HDF5 Dataset Create Property List Interface), 25
H5Pset_external (HDF5 Dataset Create Property List Interface), 25
H5Pset_fill_time (HDF5 Dataset Create Property List Interface), 25
H5Pset_fill_value (HDF5 Dataset Create Property List Interface), 25
H5Pset_filter (HDF5 Dataset Create Property List Interface), 25
H5Pset_fletcher32 (HDF5 Dataset Create Property List Interface), 25
H5Pset_layout (HDF5 Dataset Create Property List Interface), 25
H5Pset_libver_bounds (HDF5 Property List Interface), 42
H5Pset_nbit (HDF5 Dataset Create Property List Interface), 25
H5Pset_obj_track_times (HDF5 Property List Interface), 42
H5Pset_scaleoffset (HDF5 Dataset Create Property List Interface), 25
H5Pset_shuffle (HDF5 Dataset Create Property List Interface), 25
H5Pset_szip (HDF5 Dataset Create Property List Interface), 25
h5r (rhdf5), 44
h5read, *5*, *7–9*, *13–15*, *44*
h5read (h5write), 18
h5readAttributes (h5write), 18
H5S (HDF5 Dataspace Interface), 29
h5save, 14, *44*
H5Sclose, *44*
H5Sclose (HDF5 Dataspace Interface), 29
H5Scopy, *44*
H5Scopy (HDF5 Dataspace Interface), 29
H5Screate, *22*, *27*, *30*, *44*
H5Screate (HDF5 Dataspace Interface), 29
H5Screate_simple, *11*, *22*, *27*, *30*, *44*
H5Screate_simple (HDF5 Dataspace Interface), 29
h5set_extent, 15
H5Sget_simple_extent_dims, *44*
H5Sget_simple_extent_dims (HDF5 Dataspace Interface), 29
H5Sis_simple, *44*
H5Sis_simple (HDF5 Dataspace Interface), 29
H5Sselect_hyperslab, *44*
H5Sselect_hyperslab (HDF5 Dataspace Interface), 29
H5Sselect_index (HDF5 Dataspace Interface), 29
H5Sset_extent_simple (HDF5 Dataspace Interface), 29
H5Sunlimited (HDF5 Dataspace Interface), 29
H5T (HDF5 Datatype Interface), 32
H5Tcopy, *44*
H5Tcopy (HDF5 Datatype Interface), 32
h5testFileLocking, 16
H5Tget_size (HDF5 Datatype Interface), 32
H5Tget_strpad (HDF5 Datatype Interface), 32
H5Tset_size, *44*
H5Tset_size (HDF5 Datatype Interface), 32
H5Tset_strpad (HDF5 Datatype Interface), 32
h5validObjects (h5listIdentifier), 12
h5version, 18, *35*, *44*
h5write, *5*, *7–9*, *14*, *15*, 18, *44*
h5writeAttribute, *5*
h5writeAttribute (h5write), 18
h5writeDataset (h5write), 18
HDF5 (rhdf5), 44
hdf5 (rhdf5), 44
HDF5 Attribute Interface, 22
HDF5 Dataset Access Property List Interface, 24
HDF5 Dataset Create Property List Interface, 25
HDF5 Dataset Interface, 27
HDF5 Dataspace Interface, 29
HDF5 Datatype Interface, 32
HDF5 File Interface, 33

HDF5 General Library Functions, 34
HDF5 Group Interface, 35
HDF5 Identifier Interface, 37
HDF5 Link Create Property List Interface, 38
HDF5 Link Interface, 39
HDF5 Object Interface, 41
HDF5 Property List Interface, 42

rhdf5, *4, 5*, *7–12*, *14–16*, *18*, *21*, *23, 24*, *26*, *29*, *31, 32*, *34–37*, *39–41*, *43*, 44

show,H5IdComponent-method *(*H5IdComponent-class*)*, 11
suppressMessages, *6*
suppressWarnings, *6*