

# Package ‘rhdf5’

September 24, 2012

**Type** Package

**Title** HDF5 interface to R

**Version** 2.0.2

**Date** 2012-04-26

**Author** Bernd Fischer, Gregoire Pau

**Maintainer** Bernd Fischer <bernd.fischer@embl.de>

**Description** This R/Bioconductor 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

**LazyLoad** true

**Imports** zlibbioc

**Depends** methods

**SystemRequirements** GNU make

**biocViews** Infrastructure

## R topics documented:

h5const . . . . .	2
h5createAttribute . . . . .	3
h5createDataset . . . . .	4
h5createFile . . . . .	6
h5createGroup . . . . .	7
H5IdComponent-class . . . . .	8
h5listIdentifier . . . . .	8
h5ls . . . . .	9
h5save . . . . .	11
h5write . . . . .	12

HDF5 Attribute Interface . . . . .	15
HDF5 Dataset Interface . . . . .	16
HDF5 Dataspace Interface . . . . .	19
HDF5 Datatype Interface . . . . .	21
HDF5 File Interface . . . . .	22
HDF5 Group Interface . . . . .	23
HDF5 Identifier Interface . . . . .	24
HDF5 Link Interface . . . . .	25
HDF5 Object Interface . . . . .	26
rhdf5 . . . . .	27
<b>Index</b>	<b>30</b>

---

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

<http://www.hdfgroup.org/HDF5>

## See Also

[rhdf5](#)

**Examples**

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

---

h5createAttribute	<i>Create HDF5 attribute</i>
-------------------	------------------------------

---

**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)
```

**Arguments**

obj	The name (character) of the object the attribute will be attached to. For advanced programmers it is possible to provide an object of class <a href="#">H5IdComponent</a> representing a H5 object identifier (file, group, dataset). See <a href="#">H5Fcreate</a> , <a href="#">H5Fopen</a> , <a href="#">H5Gcreate</a> , <a href="#">H5Gopen</a> , <a href="#">H5Dcreate</a> , <a href="#">H5Dopen</a> 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 <a href="#">H5IdComponent</a> representig an H5 location identifier. See <a href="#">H5Fcreate</a> , <a href="#">H5Fopen</a> , <a href="#">H5Gcreate</a> , <a href="#">H5Gopen</a> to create an object of this kind. The file argument is not required, if the argument obj is of type <a href="#">H5IdComponent</a> .
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 <code>storage.mode(mydata)</code> .
H5type	Advanced programmers can specify the datatype of the dataset within the file. See <code>h5const("H5T")</code> 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.

**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**

<http://www.hdfgroup.org/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 = NULL, level = 6)
```

**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 <a href="#">H5IdComponent</a> representing a H5 location identifier (file or group). See <a href="#">H5Fcreate</a> , <a href="#">H5Fopen</a> , <a href="#">H5Gcreate</a> , <a href="#">H5Gopen</a> 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. <a href="#">HDFView</a> ), 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.
storage.mode	The storage mode of the data to be written. Can be obtained by <code>storage.mode(mydata)</code> .

H5type	Advanced programmers can specify the datatype of the dataset within the file. See <code>h5const("H5T")</code> for a list of available datatypes. If H5type is specified the argument <code>storage.mode</code> is ignored. It is recommended to use <code>storage.mode</code>
size	For <code>storage.mode='character'</code> 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.
chunk	The chunk size used to store the dataset. It is an integer vector of the same length as <code>dims</code> . This argument is usually set together with a compression property (argument <code>level</code> ).
level	The compression level used. An integer value between 0 (no compression) and 9 (highest and slowest compression).

### 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

<http://www.hdfgroup.org/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=7)

# 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")

# 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))

h5dump("ex_createDataset.h5")
```

---

h5createFile	<i>Create HDF5 file</i>
--------------	-------------------------

---

**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 if file was created successfully and FALSE otherwise.

**Author(s)**

Bernd Fischer

**References**

<http://www.hdfgroup.org/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	<i>Create HDF5 group</i>
---------------	--------------------------

---

### 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 <code>H5IdComponent</code> representing a H5 location identifier (file or group). See <a href="#">H5Fcreate</a> , <a href="#">H5Fopen</a> , <a href="#">H5Gcreate</a> , <a href="#">H5Gopen</a> 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

<http://www.hdfgroup.org/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")
```

---

H5IdComponent-class    *Class "H5IdComponent"*

---

### Description

A class representing a HDF5 identifier handle. HDF5 identifiers represent open files, groups, datasets, dataspace, 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.

### Methods

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

### Author(s)

Bernd Fischer

### References

<http://www.hdfgroup.org/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()
```



**Value**

h5validObjects returns a list of [H5IdComponent](#) objects. h5listIdentifier prints the valid identifiers on screen and returns NULL.

**Author(s)**

Bernd Fischer

**References**

<http://www.hdfgroup.org/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"))
h5dump (file,
        recursive = TRUE,
        load = TRUE,
        all = FALSE,
        index_type = h5default("H5_INDEX"),
        order = h5default("H5_ITER"), ...)
```

**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 <code>H5IdComponent</code> representing a H5 location identifier (file or group). See <code>H5Fcreate</code> , <code>H5Fopen</code> , <code>H5Gcreate</code> , <code>H5Gopen</code> 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 <code>h5const("H5_INDEX")</code> for possible arguments.
order	See <code>h5const("H5_ITER")</code> 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 <code>load=FALSE</code> and load the datasets successively.
...	Arguments passed to <code>h5read</code>

**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 dataset with the dataset header information `load=FALSE`.

**Author(s)**

Bernd Fischer

**References**

<http://www.hdfgroup.org/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)
```

## 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 <a href="#">H5IdComponent</a> representing a H5 location identifier (file or group). See <a href="#">H5Fcreate</a> , <a href="#">H5Fopen</a> , <a href="#">H5Gcreate</a> , <a href="#">H5Gopen</a> 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.

## Details

The objects will be saved to the HDF5 file. If the file does not exist 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

<http://www.hdfgroup.org/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")
```

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 = TRUE, ...)
h5write         (obj, file, name, ...)
h5write.default (obj, file, name,
                createnewfile = TRUE,
                write.attributes = FALSE, ...)
h5writeDataset (obj, h5loc, name, ...)
h5writeDataset.data.frame (obj, h5loc, name, level=7, DataFrameAsCompound = TRUE)
h5writeDataset.list (obj, h5loc, name, level=7)
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=7)
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 <code>H5IdComponent</code> representing a H5 location identifier (file or group). See <code>H5Fcreate</code> , <code>H5Fopen</code> , <code>H5Gcreate</code> , <code>H5Gopen</code> to create an object of this kind.
h5obj	An object of class <code>H5IdComponent</code> representing a H5 object identifier (file, group, or dataset). See <code>H5Fcreate</code> , <code>H5Fopen</code> , <code>H5Gcreate</code> , <code>H5Gopen</code> , <code>H5Dcreate</code> , or <code>H5Dopen</code> to create an object of this kind.
name	The name of the dataset in the HDF5 file. The name of the attribute for <code>hwriteAttribute</code> .
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 <code>NULL</code> chooses 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 <code>index</code> is not <code>NULL</code> .
stride	The stride of the hypercube. Read the introduction <a href="http://ftp.hdfgroup.org/HDF5/Tutor/phypecont.html">http://ftp.hdfgroup.org/HDF5/Tutor/phypecont.html</a> before using this argument. R behaves like Fortran in this example. This argument is ignored, if <code>index</code> is not <code>NULL</code> .
block	The block size of the hyperslab. Read the introduction <a href="http://ftp.hdfgroup.org/HDF5/Tutor/phypecont.html">http://ftp.hdfgroup.org/HDF5/Tutor/phypecont.html</a> before using this argument. R behaves like Fortran in this example. This argument is ignored, if <code>index</code> is not <code>NULL</code> .
count	The number of blocks to be written. This argument is ignored, if <code>index</code> is not <code>NULL</code> .
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 <code>h5createDataset</code> to create a dataset.
compoundAsDataFrame	If <code>true</code> , a compound datatype will be coerced to a <code>data.frame</code> . 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 <code>true</code> , a <code>data.frame</code> will be saved as a compound data type. Otherwise it is saved like a list. The advantage of saving a <code>data.frame</code> as a compound data type is that it can be read as a table from python or with a <code>struct</code> -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, <code>DataFrameAsCompound=FALSE</code> is recommended.
callGeneric	If <code>TRUE</code> a generic function <code>h5read.classname</code> 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 <code>h5read</code> depending on the class attribute. Note that <code>h5read</code> is not a S3 generic function. Dispatching is done based on the HDF5 attribute after the standard <code>h5read</code> function.
size	The length of string data type. Variable length strings are not yet supported.
createnewfile	If <code>TRUE</code> , a new file will be created if necessary.
read.attributes	(logical) If <code>TRUE</code> , the HDF5 attributes are read and attached to the respective R object.
write.attributes	(logical) If <code>TRUE</code> , all R-attributes attached to the object <code>obj</code> are written to the HDF5 file.
...	Further arguments passed to <code>H5Dread</code> .

## 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: <http://www.hdfgroup.org/HDF5/Tutor/selectsimple.html>, <http://www.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.

`h5write` returns 0 if successful.

## Author(s)

Bernd Fischer

## References

<http://www.hdfgroup.org/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(
H5Aopen_by_name (h5obj, objname = ".", name)
H5Aread       (h5attribute, buf = NULL)
H5Awrite      (h5attribute, buf)

```

**Arguments**

h5obj	An object of class <a href="#">H5IdComponent</a> representing a H5 object identifier (file, group, or dataset). See <a href="#">H5Fcreate</a> , <a href="#">H5Fopen</a> , <a href="#">H5Gcreate</a> , <a href="#">H5Gopen</a> , <a href="#">H5Dcreate</a> , or <a href="#">H5Dopen</a> to create an object of this kind.
name	The name of the attribute (character).
dtype_id	A character name of a datatype. See <code>h5const("H5T")</code> for possible datatypes. Can also be an integer representing an HDF5 datatype. Only simple datatypes are allowed for attributes.
h5space	An object of class <a href="#">H5IdComponent</a> representing a H5 dataspace. See <a href="#">H5Dget_space</a> , <a href="#">H5Screate_simple</a> , <a href="#">H5Screate</a> to create an object of this kind.
h5attribute	An object of class <a href="#">H5IdComponent</a> representing a H5 attribute as created by <a href="#">H5Acreate</a> or <a href="#">H5Aopen</a>
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 <code>h5const("H5_INDEX")</code> for possible arguments.
order	See <code>h5const("H5_ITER")</code> 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 <code>h5spaceMem</code> . 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 [http://www.hdfgroup.org/HDF5/doc/RM/RM\\_H5A.html](http://www.hdfgroup.org/HDF5/doc/RM/RM_H5A.html) 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**

<http://www.hdfgroup.org/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 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, internal1 = NULL, internal2 = 6)
H5Dopen       (h5loc, name)
H5Dclose      (h5dataset)
H5Dget_space  (h5dataset)
H5Dget_type   (h5dataset)
H5Dread       (h5dataset, h5spaceFile = NULL, h5spaceMem = NULL, buf = NULL, compoundAsDataFrame)
H5Dwrite      (h5dataset, buf, h5spaceMem = NULL, h5spaceFile = NULL)

```

**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.

**internal1, internal2** For internal usage only. Will be removed in later versions.

**Details**

Interface to the HDF5 C-library `libhdf5`. See [http://www.hdfgroup.org/HDF5/doc/RM/RM\\_H5D.html](http://www.hdfgroup.org/HDF5/doc/RM/RM_H5D.html) for further details.

**Value**

`H5Dcreate` and `H5Dopen` return an object of class [H5IdComponent](#) representing 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

<http://www.hdfgroup.org/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"))
H5Screate_simple  (dims, maxdims = dims)
H5Scopy           (h5space)
H5Sclose          (h5space)
H5Sis_simple      (h5space)
H5Sget_simple_extent_dims (h5space)
H5Sselect_hyperslab (h5space, op = h5default("H5S_SELECT"),
                    start = NULL, stride = NULL, count = NULL, block = NULL)
H5Sselect_index   (h5space, index)
```

## Arguments

type	See <code>h5const("H5S")</code> for possible types.
dims	Dimension of the dataspace. This argument is similar to the <code>dim</code> attribute of an array. When viewing the HDF5 dataset with an C-program (e.g. <code>HDFView</code> ), 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.
h5space	An object of class <code>H5IdComponent</code> representing a H5 dataspace identifier. See <code>H5Dget_space</code> , <code>H5Screate_simple</code> , <code>H5Screate</code> 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 <code>h5const("H5S_SELECT")</code> 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 <a href="http://ftp.hdfgroup.org/HDF5/Tutor/phypecont.html">http://ftp.hdfgroup.org/HDF5/Tutor/phypecont.html</a> 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 <a href="http://ftp.hdfgroup.org/HDF5/Tutor/phypecont.html">http://ftp.hdfgroup.org/HDF5/Tutor/phypecont.html</a> before using this argument. R behaves like Fortran in this example.

## Details

Interface to the HDF5 C-library libhdf5. See [http://www.hdfgroup.org/HDF5/doc/RM/RM\\_H5S.html](http://www.hdfgroup.org/HDF5/doc/RM/RM_H5S.html) for further details.

As an introduction to use hyperslabs see these tutorials: See these introductions to hyperslabs: <http://www.hdfgroup.org/HDF5/Tutor/selectsimple.html>, <http://www.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 performs 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.

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

## Author(s)

Bernd Fischer

## References

<http://www.hdfgroup.org/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 dataspace after usage to free resources
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)
```

## Arguments

<code>dtype_id</code>	A character name of a datatype. See <code>h5const("H5T")</code> for possible datatypes. Can also be an integer representing an HDF5 datatype.
<code>size</code>	The total size in bytes.

## Details

Interface to the HDF5 C-library `libhdf5`. See [http://www.hdfgroup.org/HDF5/doc/RM/RM\\_H5T.html](http://www.hdfgroup.org/HDF5/doc/RM/RM_H5T.html) for further details.

## Value

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

## Author(s)

Bernd Fischer

## References

<http://www.hdfgroup.org/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")
```

**Description**

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

**Usage**

```
H5Fcreate (name, flags = h5default("H5F_ACC"))
H5Fopen   (name, flags = h5default("H5F_ACC_RD"))
H5Fclose  (h5file)
H5Fflush  (h5file, scope = h5default("H5F_SCOPE"))
```

**Arguments**

name	The filename of the HDF5 file.
flags	See <code>h5const("H5F_ACC")</code> for possible arguments.
h5file	An object of class <code>H5IdComponent</code> representing a H5 file identifier as created by <code>H5Fcreate</code> or <code>H5Fopen</code> .
scope	See <code>h5const("H5F_ACC_RD")</code> for possible arguments.

**Details**

Interface to the HDF5 C-library `libhdf5`. See [http://www.hdfgroup.org/HDF5/doc/RM/RM\\_H5F.html](http://www.hdfgroup.org/HDF5/doc/RM/RM_H5F.html) for further details.

**Value**

`H5Fcreate` and `H5Fopen` return an object of class `H5IdComponent` representing a H5 file identifier. The other functions return the standard return value from their respective C-functions.

**Author(s)**

Bernd Fischer

**References**

<http://www.hdfgroup.org/HDF5>

**See Also**

`rhdf5`

**Examples**

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

**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 <a href="#">H5IdComponent</a> representing a H5 location identifier (file or group). See <a href="#">H5Fcreate</a> , <a href="#">H5Fopen</a> , <a href="#">H5Gcreate</a> , <a href="#">H5Gopen</a> to create an object of this kind.
name	Name of the group.
h5group	An object of class <a href="#">H5IdComponent</a> representing a H5 group identifier. See <a href="#">H5Gcreate</a> , <a href="#">H5Gopen</a> 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 <code>h5const("H5_INDEX")</code> for possible arguments.
order	See <code>h5const("H5_ITER")</code> for possible arguments.

**Details**

Interface to the HDF5 C-library `libhdf5`. See [http://www.hdfgroup.org/HDF5/doc/RM/RM\\_H5G.html](http://www.hdfgroup.org/HDF5/doc/RM/RM_H5G.html) 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

<http://www.hdfgroup.org/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 <a href="#">H5IdComponent</a> representing a H5 identifier (file, group, dataset, dataspace, datatype, attribute). See e.g. <a href="#">H5Fcreate</a> , <a href="#">H5Fopen</a> , <a href="#">H5Gcreate</a> , <a href="#">H5Gopen</a> , <a href="#">H5Dcreate</a> , <a href="#">H5Dopen</a> to create an object of this kind.
h5obj	An object of class <a href="#">H5IdComponent</a> representing a H5 object identifier (file, group, or dataset). See <a href="#">H5Fcreate</a> , <a href="#">H5Fopen</a> , <a href="#">H5Gcreate</a> , <a href="#">H5Gopen</a> , <a href="#">H5Dcreate</a> , or <a href="#">H5Dopen</a> to create an object of this kind.

## Details

Interface to the HDF5 C-library libhdf5. See [http://www.hdfgroup.org/HDF5/doc/RM/RM\\_H5I.html](http://www.hdfgroup.org/HDF5/doc/RM/RM_H5I.html) 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**<http://www.hdfgroup.org/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 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**

```
H5Lexists (h5loc, name)
H5Lget_info (h5loc, name)
```

**Arguments**

h5loc	An object of class <a href="#">H5IdComponent</a> representing a H5 location identifier (file or group). See <a href="#">H5Fcreate</a> , <a href="#">H5Fopen</a> , <a href="#">H5Gcreate</a> , <a href="#">H5Gopen</a> 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 [http://www.hdfgroup.org/HDF5/doc/RM/RM\\_H5L.html](http://www.hdfgroup.org/HDF5/doc/RM/RM_H5L.html) 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

**References**

<http://www.hdfgroup.org/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")
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 <a href="#">H5IdComponent</a> representing a H5 object identifier (file, group, or dataset). See <a href="#">H5Fcreate</a> , <a href="#">H5Fopen</a> , <a href="#">H5Gcreate</a> , <a href="#">H5Gopen</a> , <a href="#">H5Dcreate</a> , or <a href="#">H5Dopen</a> to create an object of this kind.
h5loc	An object of class <a href="#">H5IdComponent</a> representing a H5 location identifier (file or group). See <a href="#">H5Fcreate</a> , <a href="#">H5Fopen</a> , <a href="#">H5Gcreate</a> , <a href="#">H5Gopen</a> 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 [http://www.hdfgroup.org/HDF5/doc/RM/RM\\_H50.html](http://www.hdfgroup.org/HDF5/doc/RM/RM_H50.html) 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

<http://www.hdfgroup.org/HDF5>

## See Also

[rhdf5](#)

## Examples

```
# create an hdf5 file and write something
h5createFile("ex_H50.h5")
h5createGroup("ex_H50.h5", "foo")
B = array(seq(0.1, 2.0, by=0.1), dim=c(5, 2, 2))
h5write(B, "ex_H50.h5", "foo/B")

# reopen file and dataset and get object info
fid <- H5Fopen("ex_H50.h5")
oid = H5Oopen(fid, "foo")
H5Oget_num_attrs(oid)
H5Oclose(oid)
H5Fclose(fid)
```

## 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 convenient 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](#)

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

- 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](mailto:bernd.fischer@embl.de)> EMBL - European Molecular Biology Laboratory Heidelberg Germany

The package contains the HDF5 library (<http://www.hdfgroup.org/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))
```

# Index

## \*Topic **IO**

- h5const, 2
- h5createAttribute, 3
- h5createDataset, 4
- h5createFile, 6
- h5createGroup, 7
- h5listIdentifier, 8
- h5ls, 9
- h5save, 11
- h5write, 12
- HDF5 Attribute Interface, 15
- HDF5 Dataset Interface, 16
- HDF5 Dataspace Interface, 19
- HDF5 Datatype Interface, 21
- HDF5 File Interface, 22
- HDF5 Group Interface, 23
- HDF5 Identifier Interface, 24
- HDF5 Link Interface, 25
- HDF5 Object Interface, 26

## \*Topic **classes**

- H5IdComponent-class, 8

## \*Topic **file**

- h5const, 2
- h5createAttribute, 3
- h5createDataset, 4
- h5createFile, 6
- h5createGroup, 7
- h5listIdentifier, 8
- h5ls, 9
- h5save, 11
- h5write, 12
- HDF5 Attribute Interface, 15
- HDF5 Dataset Interface, 16
- HDF5 Dataspace Interface, 19
- HDF5 Datatype Interface, 21
- HDF5 File Interface, 22
- HDF5 Group Interface, 23
- HDF5 Identifier Interface, 24
- HDF5 Link Interface, 25
- HDF5 Object Interface, 26

## \*Topic **interface**

- h5const, 2
- h5createAttribute, 3

- h5createDataset, 4
- h5createFile, 6
- h5createGroup, 7
- h5listIdentifier, 8
- h5ls, 9
- h5save, 11
- h5write, 12
- HDF5 Attribute Interface, 15
- HDF5 Dataset Interface, 16
- HDF5 Dataspace Interface, 19
- HDF5 Datatype Interface, 21
- HDF5 File Interface, 22
- HDF5 Group Interface, 23
- HDF5 Identifier Interface, 24
- HDF5 Link Interface, 25
- HDF5 Object Interface, 26

## \*Topic **package**

- rhdf5, 27

## \*Topic **programming**

- h5const, 2
- h5createAttribute, 3
- h5createDataset, 4
- h5createFile, 6
- h5createGroup, 7
- h5listIdentifier, 8
- h5ls, 9
- h5save, 11
- h5write, 12
- HDF5 Attribute Interface, 15
- HDF5 Dataset Interface, 16
- HDF5 Dataspace Interface, 19
- HDF5 Datatype Interface, 21
- HDF5 File Interface, 22
- HDF5 Group Interface, 23
- HDF5 Identifier Interface, 24
- HDF5 Link Interface, 25
- HDF5 Object Interface, 26

H5 (rhdf5), 27

H5A (HDF5 Attribute Interface), 15

H5Aclose, 28

H5Aclose (HDF5 Attribute Interface), 15

H5Acreate, 8, 28

H5Acreate (HDF5 Attribute Interface), 15

- H5Adelete, 28
- H5Adelete (HDF5 Attribute Interface), 15
- H5Aexists, 28
- H5Aexists (HDF5 Attribute Interface), 15
- H5Aget\_name, 28
- H5Aget\_name (HDF5 Attribute Interface), 15
- H5Aget\_space, 28
- H5Aget\_space (HDF5 Attribute Interface), 15
- H5Aget\_type, 28
- H5Aget\_type (HDF5 Attribute Interface), 15
- H5Aopen, 8, 28
- H5Aopen (HDF5 Attribute Interface), 15
- H5Aopen\_by\_idx, 28
- H5Aopen\_by\_idx (HDF5 Attribute Interface), 15
- H5Aopen\_by\_name, 28
- H5Aopen\_by\_name (HDF5 Attribute Interface), 15
- H5Aread, 28
- H5Aread (HDF5 Attribute Interface), 15
- H5Awrite, 28
- H5Awrite (HDF5 Attribute Interface), 15
- h5const, 2, 28
- h5constType, 28
- h5constType (h5const), 2
- h5createAttribute, 3
- h5createDataset, 4, 4, 6, 7, 13, 14, 28
- h5createFile, 4, 5, 6, 7, 14, 28
- h5createGroup, 4-6, 7, 28
- H5D (HDF5 Dataset Interface), 16
- H5Dclose, 28
- H5Dclose (HDF5 Dataset Interface), 16
- H5Dcreate, 3, 8, 13, 15, 17, 24, 26, 28
- H5Dcreate (HDF5 Dataset Interface), 16
- h5default, 28
- h5default (h5const), 2
- H5Dget\_space, 8, 15, 17, 19, 28
- H5Dget\_space (HDF5 Dataset Interface), 16
- H5Dget\_type (HDF5 Dataset Interface), 16
- H5Dopen, 3, 8, 13, 15, 17, 24, 26, 28
- H5Dopen (HDF5 Dataset Interface), 16
- H5Dread, 13, 28
- H5Dread (HDF5 Dataset Interface), 16
- h5dump, 11, 28
- h5dump (h5ls), 9
- H5Dwrite, 28
- H5Dwrite (HDF5 Dataset Interface), 16
- H5F (HDF5 File Interface), 22
- H5Fclose, 28
- H5Fclose (HDF5 File Interface), 22
- H5Fcreate, 3, 4, 7, 8, 10-13, 15, 17, 23-26, 28
- H5Fcreate (HDF5 File Interface), 22
- H5Fflush, 28
- H5Fflush (HDF5 File Interface), 22
- H5Fopen, 3, 4, 7, 8, 10-13, 15, 17, 23-26, 28
- H5Fopen (HDF5 File Interface), 22
- H5G (HDF5 Group Interface), 23
- H5Gclose, 28
- H5Gclose (HDF5 Group Interface), 23
- H5Gcreate, 3, 4, 7, 8, 10-13, 15, 17, 23-26, 28
- H5Gcreate (HDF5 Group Interface), 23
- H5Gcreate\_anon, 28
- H5Gcreate\_anon (HDF5 Group Interface), 23
- H5Gget\_info, 28
- H5Gget\_info (HDF5 Group Interface), 23
- H5Gget\_info\_by\_idx, 28
- H5Gget\_info\_by\_idx (HDF5 Group Interface), 23
- H5Gget\_info\_by\_name, 28
- H5Gget\_info\_by\_name (HDF5 Group Interface), 23
- H5Gopen, 3, 4, 7, 8, 10-13, 15, 17, 23-26, 28
- H5Gopen (HDF5 Group Interface), 23
- H5I (HDF5 Identifier Interface), 24
- H5IdComponent, 3, 4, 7, 9-13, 15-17, 19, 20, 22-27
- H5IdComponent (H5IdComponent-class), 8
- H5IdComponent-class, 8
- H5Iget\_name, 28
- H5Iget\_name (HDF5 Identifier Interface), 24
- H5Iget\_type, 28
- H5Iget\_type (HDF5 Identifier Interface), 24
- H5Iis\_valid, 28
- H5Iis\_valid (HDF5 Identifier Interface), 24
- H5L (HDF5 Link Interface), 25
- H5Lexists, 28
- H5Lexists (HDF5 Link Interface), 25
- H5Lget\_info, 28
- H5Lget\_info (HDF5 Link Interface), 25
- h5listIdentifier, 8
- h5ls, 9, 11, 14, 28
- H5O (HDF5 Object Interface), 26
- H5Oclose, 28
- H5Oclose (HDF5 Object Interface), 26
- H5Oget\_num\_attrs, 28
- H5Oget\_num\_attrs (HDF5 Object

- Interface), 26
- H5Oget\_num\_attrs\_by\_name, 28
- H5Oget\_num\_attrs\_by\_name (HDF5 Object Interface), 26
- H5Oopen, 28
- H5Oopen (HDF5 Object Interface), 26
- h5r (rhdf5), 27
- h5read, 4–7, 10, 11, 28
- h5read (h5write), 12
- H5S (HDF5 Dataspace Interface), 19
- h5save, 11, 28
- H5Sclose, 28
- H5Sclose (HDF5 Dataspace Interface), 19
- H5Scopy, 28
- H5Scopy (HDF5 Dataspace Interface), 19
- H5Screate, 15, 17, 19, 28
- H5Screate (HDF5 Dataspace Interface), 19
- H5Screate\_simple, 8, 15, 17, 19, 28
- H5Screate\_simple (HDF5 Dataspace Interface), 19
- H5Sget\_simple\_extent\_dims, 28
- H5Sget\_simple\_extent\_dims (HDF5 Dataspace Interface), 19
- H5Sis\_simple, 28
- H5Sis\_simple (HDF5 Dataspace Interface), 19
- H5Sselect\_hyperslab, 28
- H5Sselect\_hyperslab (HDF5 Dataspace Interface), 19
- H5Sselect\_index (HDF5 Dataspace Interface), 19
- H5T (HDF5 Datatype Interface), 21
- H5Tcopy, 28
- H5Tcopy (HDF5 Datatype Interface), 21
- H5Tset\_size, 28
- H5Tset\_size (HDF5 Datatype Interface), 21
- h5validObjects (h5listIdentifier), 8
- h5write, 4–7, 10, 11, 12, 28
- h5writeAttribute, 3
- h5writeAttribute (h5write), 12
- h5writeDataset (h5write), 12
- HDF5 (rhdf5), 27
- hdf5 (rhdf5), 27
- HDF5 Attribute Interface, 15
- HDF5 Dataset Interface, 16
- HDF5 Dataspace Interface, 19
- HDF5 Datatype Interface, 21
- HDF5 File Interface, 22
- HDF5 Group Interface, 23
- HDF5 Identifier Interface, 24
- HDF5 Link Interface, 25
- HDF5 Object Interface, 26
- rhdf5, 2, 4–11, 14, 16, 18, 20–22, 24–27, 27
- show, H5IdComponent-method (H5IdComponent-class), 8