

# Package ‘rhdf5’

November 29, 2022

**Type** Package

**Title** R Interface to HDF5

**Version** 2.42.0

**Date** 2022-10-26

**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 (>= 1.13.4), rhdf5filters

**Depends** R (>= 4.0.0), methods

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

**LinkingTo** Rhdf5lib

**SystemRequirements** GNU make

**biocViews** Infrastructure, DataImport

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.1

**git\_url** <https://git.bioconductor.org/packages/rhdf5>

**git\_branch** RELEASE\_3\_16

**git\_last\_commit** fa26027

**git\_last\_commit\_date** 2022-11-01

**Date/Publication** 2022-11-29

**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:

H5Aclose . . . . .	5
H5Acreate . . . . .	5
H5Adelete . . . . .	6
H5Aexists . . . . .	6
H5Aget_name . . . . .	7
H5Aget_space . . . . .	7
H5Aget_type . . . . .	8
H5Aopen . . . . .	8
H5Aread . . . . .	9
H5Awrite . . . . .	10
h5closeAll . . . . .	10
h5constants . . . . .	11
H5Dchunk_dims . . . . .	12
H5Dclose . . . . .	12
H5Dcreate . . . . .	13
H5Dget_create_plist . . . . .	14
H5Dget_space . . . . .	14
H5Dget_storage_size . . . . .	15
H5Dget_type . . . . .	15
H5Dopen . . . . .	16
H5Dread . . . . .	17
H5Dset_extent . . . . .	18
H5Dwrite . . . . .	19
H5Fclose . . . . .	19
H5Fcreate . . . . .	20
H5Fflush . . . . .	20
H5Fget_filesize . . . . .	21
H5Fget_name . . . . .	21
H5Fget_plist . . . . .	22
H5Fis_hdf5 . . . . .	23
H5Fopen . . . . .	23
H5functions . . . . .	24
H5Gclose . . . . .	25
H5Gcreate . . . . .	25
H5Gcreate_anon . . . . .	26
H5Gget_info . . . . .	26

H5Gopen	27
H5IdComponent-class	28
H5Iget_name	29
H5Iget_type	30
H5lis_valid	31
H5Lcopy	31
H5Lcreate_external	32
H5Ldelete	33
H5Lexists	34
H5Lget_info	34
h5listObjects	35
H5Lmove	36
h5ls	37
H5Oclose	38
H5Ocopy	39
H5Oget_num_attrs	40
H5Olink	41
H5Oopen	42
H5Pall_filters_avail	43
H5Pclose	43
H5Pcopy	44
H5Pcreate	44
H5Pfill_value_defined	45
H5Pget_class	45
H5Pget_version	46
H5Pobject_track_times	46
H5Pset_blosc	47
H5Pset_bzip2	47
H5Pset_deflate	48
H5Pset_fapl_ros3	48
H5Pset_istore_k	49
H5Pset_lzf	49
H5Pset_nbit	50
H5Pset_shared_mesg_index	50
H5Pset_shared_mesg_nindexes	51
H5Pset_shared_mesg_phase_change	51
H5Pset_shuffle	52
H5Pset_sizes	52
H5Pset_sym_k	53
H5Pset_szip	53
H5Pset_userblock	54
H5P_chunk	54
H5P_chunk_cache	55
H5P_create_intermediate_group	55
H5P_fill_time	56
H5P_fill_value	56
H5P_layout	57
H5P_libver_bounds	58

H5R	58
H5Rcreate	59
H5Rdereference	60
H5Ref-class	60
H5Rget_name	61
H5Rget_obj_type	62
H5Rget_region	62
H5Sclose	63
H5Scombine_hyperslab	63
H5Scombine_select	64
H5Scopy	65
H5Screate	66
H5Screate_simple	67
H5Sget_select_npoints	67
H5Sget_simple_extent_dims	68
H5Sis_simple	68
H5Sselect_all	68
H5Sselect_hyperslab	69
H5Sselect_index	70
H5Sselect_none	71
H5Sselect_valid	71
H5Sset_extent_simple	72
H5Sunlimited	72
H5Tcopy	73
H5Tis_variable_str	73
H5T_cset	74
H5T_precision	74
H5T_size	75
H5T_strpad	75
h5version	76
H5Zfilter_avail	76
h5_createAttribute	77
h5_createDataset	79
h5_createFile	82
h5_createGroup	83
h5_delete	84
h5_deleteAttribute	85
h5_dump	85
h5_errorHandling	87
h5_FileLocking	88
h5_read	89
h5_readAttributes	92
h5_save	92
h5_set_extent	93
h5_write	94
h5_writeAttribute	98
rhdf5	99

---

H5Aclose	<i>Close an HDF5 attribute</i>
----------	--------------------------------

---

**Description**

Close an HDF5 attribute

**Usage**

```
H5Aclose(h5attribute)
```

**Arguments**

h5attribute	An object of class <a href="#">H5IdComponent</a> representing a the attribute to be closed. Normally created by <a href="#">H5Aopen()</a> or similar.
-------------	---

**See Also**

[H5Aopen\(\)](#)

---

H5Acreate	<i>Create an attribute for an HDF5 object</i>
-----------	---

---

**Description**

Creates an attribute, name, which is attached to the object specified by the identifier h5obj. The attribute name must be unique for the object.

**Usage**

```
H5Acreate(h5obj, name, dtype_id, h5space)
```

**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.

**Value**

An object of class [H5IdComponent](#) representing a H5 attribute identifier.

H5Adelete

*Delete an specified attribute of an HDF5 object*

---

**Description**

Delete an specified attribute of an HDF5 object

**Usage**

```
H5Adelete(h5obj, 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.
name	The name of the attribute (character).

---

H5Aexists*Check whether an specific attribute exists for an HDF5 object*

---

**Description**

Check whether an specific attribute exists for an HDF5 object

**Usage**

```
H5Aexists(h5obj, 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.
name	The name of the attribute (character).

---

H5Aget_name	<i>Get the name of an HDF5 attribute object</i>
-------------	---

---

**Description**

Retrieves the name of the attribute specified by an HDF5 attribute object.

**Usage**

```
H5Aget_name(h5attribute)
```

**Arguments**

h5attribute     An object of class [H5IdComponent](#) representing an attribute. Normally created by [H5Aopen\(\)](#) or similar.

**Value**

A character vector of length 1 containing the name of the attribute.

---

H5Aget_space	<i>Get a copy of the attribute dataspace</i>
--------------	--

---

**Description**

Get a copy of the attribute dataspace

**Usage**

```
H5Aget_space(h5attribute)
```

**Arguments**

h5attribute     An object of class [H5IdComponent](#) representing an attribute. Normally created by [H5Aopen\(\)](#) or similar.

**Value**

Returns an object of class [H5IdComponent](#) representing a H5 dataspace identifier

---

H5Aget_type	<i>Get a copy of the attribute datatype</i>
-------------	---

---

### Description

Get a copy of the attribute datatype

### Usage

```
H5Aget_type(h5attribute)
```

### Arguments

h5attribute	An object of class <a href="#">H5IdComponent</a> representing an attribute. Normally created by <a href="#">H5Aopen()</a> or similar.
-------------	---

---

H5Aopen	<i>Open an attribute for an HDF5 object</i>
---------	---

---

### Description

Open an attribute for an HDF5 object

### Usage

```
H5Aopen(h5obj, name)
```

```
H5Aopen_by_name(h5obj, objname = ".", name)
```

```
H5Aopen_by_idx(
    h5obj,
    n,
    objname = ".",
    index_type = h5default("H5_INDEX"),
    order = h5default("H5_ITER")
)
```

### 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).
objname	The name of the object the attribute belongs to.



n	Opens attribute number n in the given order and index. Indexing is C-style, base-0, so the first attribute is opened with n=0.
index_type	See <code>h5const("H5_INDEX")</code> for possible arguments.
order	See <code>h5const("H5_ITER")</code> for possible arguments.

**Value**

An object of class [H5IdComponent](#) representing a H5 attribute identifier.

---

H5Aread	<i>Read data from an HDF5 attribute</i>
---------	---

---

**Description**

Read data from an HDF5 attribute

**Usage**

```
H5Aread(h5attribute, buf = NULL, bit64conversion)
```

**Arguments**

h5attribute	An object of class <a href="#">H5IdComponent</a> representing an attribute. Normally created by <a href="#">H5Aopen()</a> or similar.
buf	Optional buffer to store retrieved values. The buffer size has to fit the size of the memory space <code>h5spaceMem</code> . No extra memory will be allocated for the data. Default is NULL which means the function will return the attribute data.
bit64conversion	Defines how 64-bit integers are converted. (See the details section for more information on these options.)

**Details**

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 risk 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.

**Value**

If `buf=NULL` returns the contents of the attribute. Otherwise return 0 if attribute is read successfully.

---

H5Awrite	<i>Write data to an HDF5 attribute</i>
----------	--

---

**Description**

Write data to an HDF5 attribute

**Usage**

```
H5Awrite(h5attribute, buf)
```

**Arguments**

h5attribute	An object of class <a href="#">H5IdComponent</a> representing an attribute. Normally created by <a href="#">H5Aopen()</a> or similar.
buf	The data to be written.

---

h5closeAll	<i>Close all open HDF5 handles</i>
------------	------------------------------------

---

**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()
```

**Value**

Doesn't return anything. Called for the side-effect of closing any open HDF5 handles.

**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()
```

---

h5constants	<i>HDF5 library constants.</i>
-------------	--------------------------------

---

**Description**

Access to HDF5 constants.

**Usage**

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

**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

**Examples**

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

---

H5Dchunk_dims	<i>Return the dimensions of a dataset chunk</i>
---------------	---

---

**Description**

Return the dimensions of a dataset chunk

**Usage**

```
H5Dchunk_dims(h5dataset)
```

**Arguments**

h5dataset      Object of class [H5IdComponent](#) representing an open HDF5 dataset.

**Details**

This function does not map directly to the HDF5 C API but is included as a useful addition.

**Value**

If the supplied dataset is chunked returns a vector, with length equal to the rank of the dataset, containing the size of the dataset dimensions. Returns NULL if the given dataset is not chunked.

**Author(s)**

Mike Smith

---

H5Dclose	<i>Close an open HDF5 dataset</i>
----------	-----------------------------------

---

**Description**

Close an open HDF5 dataset

**Usage**

```
H5Dclose(h5dataset)
```

**Arguments**

h5dataset      Object of class [H5IdComponent](#) representing an open HDF5 dataset

---

H5Dcreate	<i>Create a new HDF5 dataset</i>
-----------	----------------------------------

---

### Description

Create a new HDF5 dataset

### Usage

```
H5Dcreate(
    h5loc,
    name,
    dtype_id,
    h5space,
    lcp1 = NULL,
    dcpl = NULL,
    dap1 = NULL
)
```

### 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 dataset.
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.
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
lcp1, dcpl, dap1	An objects of class <a href="#">H5IdComponent</a> representing HDF5 property lists. Specially these should respectively be: a link creation property list, a dataset creation property list, a dataset access property list

### Value

An object of class [H5IdComponent](#) representing the opened dataset.

---

H5Dget\_create\_plist     *Return a copy of the dataset creation property list for a dataset*

---

**Description**

Return a copy of the dataset creation property list for a dataset

**Usage**

```
H5Dget_create_plist(h5dataset)
```

**Arguments**

h5dataset     Object of class [H5IdComponent](#) representing an open HDF5 dataset

---

H5Dget\_space     *Return a copy of the HDF5 dataspace for a dataset*

---

**Description**

Return a copy of the HDF5 dataspace for a dataset

**Usage**

```
H5Dget_space(h5dataset)
```

**Arguments**

h5dataset     Object of class [H5IdComponent](#) representing an open HDF5 dataset

**Value**

Returns an object of class [H5IdComponent](#) representing a HDF5 dataspace identifier

---

H5Dget\_storage\_size     *Find the amount of storage allocated for a dataset*

---

**Description**

H5Dget\_storage\_size returns the amount of storage, in bytes, allocated in an HDF5 file to hold a given dataset. This is the amount of space required on-disk, which not typically a good indicator of the amount of memory that will be required to read the complete dataset.

**Usage**

```
H5Dget_storage_size(h5dataset)
```

**Arguments**

h5dataset     Object of class [H5IdComponent](#) representing an open HDF5 dataset

**Value**

Returns an integer giving the number of bytes allocated in the file to the dataset.

---

H5Dget\_type     *Return a copy of the HDF5 datatype for a dataset*

---

**Description**

Return a copy of the HDF5 datatype for a dataset

**Usage**

```
H5Dget_type(h5dataset)
```

**Arguments**

h5dataset     Object of class [H5IdComponent](#) representing an open HDF5 dataset

---

H5Dopen	<i>Open an existing HDF5 dataset</i>
---------	--------------------------------------

---

## Description

Open an existing HDF5 dataset

## Usage

```
H5Dopen(h5loc, name, dapl = NULL)
```

## Arguments

h5loc	An object of class <a href="#">H5IdComponent</a> representing a H5 location identifier (file or group).
name	Name of the dataset to open.
dapl	An object of class <a href="#">H5IdComponent</a> representing a H5 dataset access property list.

## Value

An object of class [H5IdComponent](#) representing the opened dataset. To prevent memory leaks this must be closed with a call to [H5Dclose\(\)](#) when no longer needed.

## Examples

```
h5file <- tempfile(fileext = ".h5")
h5createFile( h5file )
h5createDataset( h5file, dataset = "A", dims = 10)

fid <- H5Fopen( h5file )
did <- H5Dopen( h5loc = fid, name = "A")
did

## rember to close open handles
H5Dclose( did )
H5Fclose( fid )
```



---

H5Dread

*Read from an HDF5 dataset*


---

### Description

H5Dread() reads a (partial) dataset from an HDF5 file into the R session.

### Usage

```
H5Dread(
  h5dataset,
  h5spaceFile = NULL,
  h5spaceMem = NULL,
  buf = NULL,
  compoundAsDataFrame = TRUE,
  bit64conversion,
  drop = FALSE
)
```

### Arguments

h5dataset	Object of class <a href="#">H5IdComponent</a> representing an open HDF5 dataset.
h5spaceFile	An object of class <a href="#">H5IdComponent</a> representing a HDF5 dataspace. See <a href="#">H5Dget_space()</a> , <a href="#">H5Screate_simple()</a> , <a href="#">H5Screate()</a> to create an object of this kind.
h5spaceMem	An object of class <a href="#">H5IdComponent</a> representing a HDF5 dataspace. See <a href="#">H5Dget_space()</a> , <a href="#">H5Screate_simple()</a> , <a href="#">H5Screate()</a> 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. <a href="#">HDFView</a> ) the order of dimensions will invert, because in C the fastest changing dimension is the last one.
buf	Buffer to hold the read data. 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	Logical vector of length 1. If TRUE, 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 <code>list</code> . Nested compound data types will be returned as a nested <code>list</code> .
bit64conversion	Defines how 64-bit integers are converted. (See the details section for more information on these options.)
drop	Logical vector of length 1. If TRUE, the HDF5 object is read as a vector with NULL dim attributes. Default is FALSE.

**Details**

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 risk 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.

---

H5Dset\_extent

*Change the dimensions of an HDF5 dataset*


---

**Description**

Change the dimensions of an HDF5 dataset

**Usage**

```
H5Dset_extent(h5dataset, size)
```

**Arguments**

<code>h5dataset</code>	Object of class <a href="#">H5IdComponent</a> representing an open HDF5 dataset.
<code>size</code>	An integer vector with the new dimension of the dataset.

**Details**

This function can only be applied to datasets that meet the following criteria:

- A chunked dataset with unlimited dimensions
- A chunked dataset with fixed dimensions if the new dimension sizes are less than the maximum sizes set with `maxdims #'`

**Author(s)**

Bernd Fischer, Mike Smith

---

H5Dwrite	<i>Write data to dataset</i>
----------	------------------------------

---

**Description**

Write data to dataset

**Usage**

```
H5Dwrite(h5dataset, buf, h5spaceMem = NULL, h5spaceFile = NULL)
```

**Arguments**

`h5dataset` Object of class [H5IdComponent](#) representing an open HDF5 dataset.

`buf` The R object containing the data to be written to the dataset.

`h5spaceMem`, `h5spaceFile` [H5IdComponent](#) objects representing the memory and file dataspace respectively. If these are left NULL dataspace that match the size and shape of `h5dataset` will be used.

---

H5Fclose	<i>Close access to an HDF5 file</i>
----------	-------------------------------------

---

**Description**

Close access to an HDF5 file

**Usage**

```
H5Fclose(h5file)
```

**Arguments**

`h5file` [H5IdComponent](#) representing an HDF5 file ID. Typically created via [H5Fcreate\(\)](#) or [H5Fopen\(\)](#).

---

H5Fcreate	<i>Create an HDF5 file</i>
-----------	----------------------------

---

### Description

Create an HDF5 file

### Usage

```
H5Fcreate(
  name,
  flags = h5default("H5F_ACC"),
  fcpl = NULL,
  fapl = NULL,
  native = FALSE
)
```

### Arguments

name	The name of the HDF5 file to create.
flags	See <code>h5const("H5F_ACC")</code> for possible arguments.
fcpl, fapl	Object object of class <a href="#">H5IdComponent</a> . This should represent a file creation property list and a file access property list respectively. See <a href="#">H5Pcreate()</a> or <a href="#">H5Pcopy()</a> to create objects of this kind. Leaving as NULL will use the default HDF5 settings which are often sufficient.
native	An object of class <code>logical</code> . If TRUE, array-like objects are treated as stored in HDF5 row-major rather than R column-major orientation. Using <code>native = TRUE</code> increases HDF5 file portability between programming languages. A file written with <code>native = TRUE</code> should also be read with <code>native = TRUE</code> .

---

H5Fflush	<i>Flush all buffers associated with a file to disk</i>
----------	---

---

### Description

Flush all buffers associated with a file to disk

### Usage

```
H5Fflush(h5file, scope = h5default("H5F_SCOPE"))
```

**Arguments**

h5file	<a href="#">H5IdComponent</a> representing any object associated with the file to be flushed.
scope	Specifies whether the scope of the flushing action is global (flushes the entire virtual file) or local (flushes only the specified file). Valid values are H5F_SCOPE_GLOBAL and H5F_SCOPE_LOCAL.

---

H5Fget_filesize	<i>Find the size of an open HDF5 file</i>
-----------------	---

---

**Description**

H5Fget\_filesize() returns the size in bytes of the HDF5 file specified by h5file.

**Usage**

```
H5Fget_filesize(h5file)
```

**Arguments**

h5file	<a href="#">H5IdComponent</a> representing an HDF5 file ID. Typically created via <a href="#">H5Fcreate()</a> or <a href="#">H5Fopen()</a> .
--------	--

---

H5Fget_name	<i>Retrieve the name of the file to which an object belongs</i>
-------------	---

---

**Description**

Retrieve the name of the file to which an object belongs

**Usage**

```
H5Fget_name(h5obj)
```

**Arguments**

h5obj	An object of class <a href="#">H5IdComponent</a> . Despite this being an H5F function, it works equally well on H5 file, group, dataset and attribute datatypes.
-------	--

## Examples

```
## use an example file and show its location
h5file <- system.file("testfiles", "h5ex_t_array.h5", package = "rhdf5")
h5file

## open a file handle and confirm we can identify the file it points to
fid <- H5Fopen(h5file)
H5Fget_name(fid)

## H5Fget_name() can be applied to group and dataset handles too
gid <- H5Gopen(fid, name = "/")
did <- H5Dopen(fid, name = "DS1")
H5Fget_name(gid)
H5Fget_name(did)

## tidy up
H5Dclose(did)
H5Gclose(gid)
H5Fclose(fid)
```

---

H5Fget\_plist

*Get property lists associated with an HDF5 file*

---

## Description

Get property lists associated with an HDF5 file

## Usage

```
H5Fget_create_plist(h5file)
```

```
H5Fget_access_plist(h5file)
```

## Arguments

**h5file** An object of class [H5IdComponent](#) representing a H5 file identifier. Typically produced by [H5Fopen\(\)](#) or [H5Fcreate\(\)](#).

---

H5Fis_hdf5	<i>Determine whether a file is in the HDF5 format</i>
------------	---

---

**Description**

H5Fis\_hdf5() determines whether a file is in the HDF5 format.

**Usage**

```
H5Fis_hdf5(name, showWarnings = TRUE)
```

**Arguments**

name	Character vector of length 1, giving the path to the file to be checked.
showWarnings	If the file doesn't exist an warning is generated. Setting this argument to FALSE will suppress the warning.

**Value**

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

---

H5Fopen	<i>Open an existing HDF5 file</i>
---------	-----------------------------------

---

**Description**

Open an existing HDF5 file

**Usage**

```
H5Fopen(name, flags = h5default("H5F_ACC_RD"), fapl = NULL, native = FALSE)
```

**Arguments**

name	The name (or path) of the HDF5 file to be opened.
flags	Character string defining the access mode for opening the file.
fapl	<a href="#">H5IdComponent</a> object representing a file access property list. Leaving this argument as NULL will use the default HDF5 properties.
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 opened for reading with native = TRUE.

**Details**

Possible values for the `flags` argument are `H5F_ACC_RDWR` and `H5F_ACC_RDONLY`. Note that HDF5's "Single Write Multiple Reader (SWMR) mode is not currently supported via **rhdf5**.

---

H5functions

*HDF5 General Library Functions*

---

**Description**

These low level functions provide general library functions for HDF5.

**Usage**

`H5open()`

`H5close()`

`H5garbage_collect()`

`H5get_libversion()`

**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, Mike Smith

**Examples**

```
## Not run:  
H5open()  
H5close()  
H5garbage_collect()  
H5get_libversion()  
  
## End(Not run)
```



---

H5Gclose	<i>Close a specified group</i>
----------	--------------------------------

---

**Description**

Close a specified group

**Usage**

```
H5Gclose(h5group)
```

**Arguments**

h5group	An object of class <a href="#">H5IdComponent</a> representing a H5 group. Typically created via <a href="#">H5Gopen()</a> or <a href="#">H5Gcreate()</a> .
---------	--

---

H5Gcreate	<i>Create a new HDF5 group and link it to a location in a file</i>
-----------	--

---

**Description**

H5Gcreate is used to a new group and link it into a file.

**Usage**

```
H5Gcreate(h5loc, name)
```

**Arguments**

h5loc	An object of class <a href="#">H5IdComponent</a>
name	Name of the new group to be created.

---

H5Gcreate\_anon                      *Create a new HDF5 group without linking it into a file*

---

### Description

Create a new HDF5 group without linking it into a file

### Usage

```
H5Gcreate_anon(h5loc)
```

### Arguments

h5loc                      An object of class [H5IdComponent](#) specifying the file in which the new group is to be created.

### Value

H5Gcreate\_anon returns an object of class [H5IdComponent](#) representing the newly created group. However at this point it is still anonymous, and must be linked into the file structure via [H5Olink\(\)](#). If this is not done, the group will be deleted from the file when it is closed.

### See Also

[H5Gcreate\(\)](#), [H5Olink\(\)](#)

---

H5Gget\_info                      *Retrieve information about a group*

---

### Description

Retrieve information about a group

### Usage

```
H5Gget_info(h5loc)

H5Gget_info_by_name(h5loc, group_name)

H5Gget_info_by_idx(
    h5loc,
    n,
    group_name = ".",
    index_type = h5default("H5_INDEX"),
    order = h5default("H5_ITER")
)
```

**Arguments**

h5loc	An object of class <a href="#">H5IdComponent</a> representing a H5 group.
group_name	An additional group name specifying the group for which information is sought. It is interpreted relative to h5loc.
n	Position in the index of the group for which information is retrieved.
index_type	See <code>h5const("H5_INDEX")</code> for possible arguments.
order	See <code>h5const("H5_ITER")</code> for possible arguments.

**Value**

A list with group information

**Examples**

```
h5file <- system.file("testfiles", "multiple_dtypes.h5", package="rhdf5")
fid <- H5Fopen(h5file)
gid <- H5Gopen(fid, "/foo")
gid
H5Gget_info(gid)
H5Gclose(gid)

## the "get_info_by" functions take the H5 object that contains the
## group(s) of interest. We can retrieve information by index or by name
H5Gget_info_by_idx(fid, 3)
H5Gget_info_by_name(fid, "/foo")

H5Fclose(fid)
```

---

H5Gopen	<i>Open a specified group</i>
---------	-------------------------------

---

**Description**

Open a specified group

**Usage**

```
H5Gopen(h5loc, name)
```

**Arguments**

h5loc	An object of class <a href="#">H5IdComponent</a> representing a H5 file or group that contains the group to be opened.
name	Name of the group to open.

**Value**

An object of class [H5IdComponent](#) representing the opened group. When access to the group is no longer needed this should be released with [H5Gclose\(\)](#) to prevent resource leakage.

**See Also**

[H5Gclose\(\)](#)

---

H5IdComponent-class    *An S4 class representing an H5 object*

---

**Description**

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

**Usage**

```
## S4 method for signature 'H5IdComponent'
show(object)

## S4 method for signature 'H5IdComponent,character'
e1 & e2

## S4 method for signature 'H5IdComponent'
x$name

## S4 replacement method for signature 'H5IdComponent'
x$name <- value

## S4 method for signature 'H5IdComponent'
x[i, j, ..., drop = TRUE]

## S4 replacement method for signature 'H5IdComponent'
x[i, j, ...] <- value
```

**Arguments**

object	Object of class H5IdComponent
e1	An H5IdComponent object representing an H5 file or group.
e2	Character giving the path to an HDF5 group or dataset relative to e1.
x	Object of class H5IdComponent representing the HDF5 dataset from which to extract element(s) or in which to replace element(s).
name	Character giving the path to an HDF5 group or dataset relative to x.
value	Array-like R object containing value to be inserted into the HDF5 dataset.

<code>i, j, ...</code>	Indices specifying elements to extract or replace. Indices are numeric vectors or empty (missing) or NULL. Numeric values are coerced to integer as by <a href="#">as.integer</a> (and hence truncated towards zero).
<code>drop</code>	If TRUE the result is coerced to the lowest possible dimension (see the examples). This only works for extracting elements, not for the replacement. See <a href="#">drop</a> for further details.

### Methods (by generic)

- `show(H5IdComponent)`: Print details of the object to screen.
- `e1 & e2`: Returns a group handle or dataset handle for the group or dataset name in the HDF5 location `h5loc`. `h5loc` 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'`.
- `$`: Reads the HDF5 object name in the HDF5 location `x`. `x` can either be a file handle as returned by [H5Fopen](#) or a group handle as e.g. returned by `h5f$g1` or `h5f$'/g1/g2'`.
- ``$` (H5IdComponent) <- value`: 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.
- `[`: 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.
- ``[` (H5IdComponent) <- value`: 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](#).

### Slots

`ID` integer of length 1. 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`

---

H5Iget\_name

*Retrieve the name of an object from a given identifier*

---

### Description

Retrieve the name of an object from a given identifier

### Usage

`H5Iget_name(h5obj)`

**Arguments**

h5obj            An object of class [H5IdComponent](#). Can represent a file, group, dataset or attribute.

---

H5Iget\_type            *Find the type of an object*

---

**Description**

Possible types returned by the function are:

- H5I\_FILE
- H5I\_GROUP
- H5I\_DATATYPE
- H5I\_DATASPACE
- H5I\_DATASET
- H5I\_ATTR

**Usage**

```
H5Iget_type(h5identifier)
```

**Arguments**

h5identifier    Object of class [H5IdComponent](#).

**Value**

Returns a character vector of length 1 containing the HDF5 type for the supplied identifier.

**Examples**

```
h5file <- system.file("testfiles", "h5ex_t_array.h5", package="rhdf5")
fid <- H5Fopen(h5file)
gid <- H5Gopen(fid, "/")

## identify the HDF5 types for these identifiers
H5Iget_type(fid)
H5Iget_type(gid)

## tidy up
H5Gclose(gid)
H5Fclose(fid)
```

---

H5Iis_valid	<i>Determine whether an identifier is valid</i>
-------------	---

---

**Description**

An identifier is no longer valid after it has been closed.

**Usage**

```
H5Iis_valid(h5identifier)
```

**Arguments**

h5identifier    Object of class [H5IdComponent](#).

**Value**

A logical of length 1. TRUE is the identifier is valid, FALSE if not.

**Examples**

```
h5file <- system.file("testfiles", "h5ex_t_array.h5", package="rhdf5")
fid <- H5Fopen(h5file)

## test whether the identifier to the opened file is valid
H5Iis_valid(fid)

## the file ID is no longer valid after it has been closed
H5Fclose(fid)
H5Iis_valid(fid)
```

---

H5Lcopy	<i>Copy a link from one location to another</i>
---------	---

---

**Description**

Copy a link from one location to another

**Usage**

```
H5Lcopy(h5loc, name, h5loc_dest, name_dest, lcpl = NULL, lapl = NULL)
```

**Arguments**

h5loc	An object of class <a href="#">H5IdComponent</a> representing a H5 location identifier (file or group) where the new link is placed.
name	The name of the link to be copied.
h5loc_dest	An object of class <a href="#">H5IdComponent</a> representing the destination file or group where a copied or moved link should be created.
name_dest	The name of the link to be created when copying or moving.
lcp1, lap1	Link creation and link access property lists. If left as NULL the HDF5 defaults will be used.

---

H5Lcreate\_external      *Create a link to an object in a different HDF5 file*

---

**Description**

H5Lcreate\_external() creates a new external link. An external link is a soft link to an object in a different HDF5 file from the location of the link.

**Usage**

```
H5Lcreate_external(target_file_name, target_obj_name, link_loc, link_name)
```

**Arguments**

target_file_name	Name of the external HDF5 to link to
target_obj_name	Path to the object in the file specified by target_file_name to link to.
link_loc	<a href="#">H5IdComponent</a> object giving the location where the new link should be created. Can represent an HDF5 file or group.
link_name	Name (path) of the new link, relative to the location of link_loc.

**Examples**

```
## The example below creates a new HDF5 file in a temporary director, and then
## links to the group "/foo" found in the file "multiple_dtypes.h5"
## distributed with the package.

h5File1 <- system.file("testfiles", "multiple_dtypes.h5", package="rhdf5")
h5File2 <- tempfile(pattern = "H5L_2_", fileext = ".h5")
h5createFile(h5File2)

## open the new file & create a link to the group "/foo" in the original file
fid <- H5Fopen(h5File2)
H5Lcreate_external(target_file_name = h5File1, target_obj_name = "/foo",
  link_loc = fid, link_name = "/external_link")
```



```
H5Fclose(fid)

## check the new file has a group called "/external_link"
h5ls(h5File2)
```

---

H5Ldelete	<i>Remove a link from a group</i>
-----------	-----------------------------------

---

### Description

Remove a link from a group

### Usage

```
H5Ldelete(h5loc, name)
```

### Arguments

h5loc	An object of class <a href="#">H5IdComponent</a> representing a H5 location identifier (file or group).
name	The name of the link to be deleted.

### Examples

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

# reopen file and confirm "/foo" exists but "/baa" does not
fid <- H5Fopen("ex_H5L.h5")
H5Lexists(fid, "/foo")

# remove the link to "/foo" and confirm it no longer exists
H5Ldelete(fid, "/foo")
H5Lexists(fid, "/foo")

H5Fclose(fid)
```

---

H5Lexists	<i>Confirm existence of a link</i>
-----------	------------------------------------

---

**Description**

Confirm existence of a link

**Usage**

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

**Arguments**

h5loc	An object of class <a href="#">H5IdComponent</a> representing a H5 location identifier (file or group).
name	The name of the link to be checked

---

H5Lget_info	<i>Find information about a link</i>
-------------	--------------------------------------

---

**Description**

H5Lget\_info() identifies the type of link specified by the the h5loc and name arguments. This is more limited than the equivalent function in the standard HDF5 library.

**Usage**

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

**Arguments**

h5loc	An object of class <a href="#">H5IdComponent</a> representing a H5 location identifier (file or group).
name	The name of the link to be queried.

**Value**

A character vector of length 1 giving the type of link. Possible values are: H5L\_TYPE\_HARD, H5L\_TYPE\_SOFT, H5L\_TYPE\_EXTERNAL, H5L\_TYPE\_ERROR

---

h5listObjects	<i>List all open HDF5 objects.</i>
---------------	------------------------------------

---

**Description**

A list of all valid HDF5 identifiers. 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, Mike Smith

**Examples**

```
h5createFile("ex_list_identifier.h5")

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

h5listIdentifier()
h5validObjects()
```

H5Lmove

*Move a link within an HDF5 file***Description**

Move a link within an HDF5 file

**Usage**

```
H5Lmove(h5loc, name, h5loc_dest, name_dest, lcpl = NULL, lapl = NULL)
```

**Arguments**

h5loc	An object of class <a href="#">H5IdComponent</a> representing a H5 location identifier (file or group) where the new link is placed.
name	The name of the link to be moved.
h5loc_dest	<a href="#">H5IdComponent</a> object representing the H5 location where the new link should be created.
name_dest	Name of the new link to be created
lcpl, lapl	Link creation and link access property lists to be associated with the new link. Leaving these arguments as NULL will use the HDF5 default property lists.

**Examples**

```
## create an HDF5 file with a single group
## that contains a dataset of 10 numbers
h5file <- tempfile(fileext = ".h5")
h5createFile(h5file)
h5createGroup(h5file, "/foo")
h5write(1:10, h5file, name = "/foo/vector1")
## check the structure is what we expect
h5ls(h5file)

## open the file, the group where the dataset currently is
## and the root group
fid <- H5Fopen(name = h5file)
gid1 <- H5Gopen(fid, "/foo")
gid2 <- H5Gopen(fid, "/")
## move the dataset to the root of the file and rename it
H5Lmove(gid1, "vector1", gid2, "vector_new")
h5closeAll()
## check the dataset has moved out of the foo group
h5ls(h5file)

## we can also provide the ID of the HDF5 file
## and use the "name" arguments to move between groups
fid <- H5Fopen(name = h5file)
H5Lmove(fid, "/vector_new", fid, "/foo/vector_newer")
```

```
H5Fclose(fid)
h5ls(h5file)
```

---

h5ls *List the content of an HDF5 file.*

---

### Description

List the content of an HDF5 file.

### Usage

```
h5ls(
  file,
  recursive = TRUE,
  all = FALSE,
  datasetinfo = TRUE,
  index_type = h5default("H5_INDEX"),
  order = h5default("H5_ITER"),
  s3 = FALSE,
  s3credentials = NULL,
  native = FALSE
)
```

### Arguments

file	The filename (character) of the file in which the dataset will be located. You can also 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.
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.
s3	Logical value indicating whether the file argument should be treated as a URL to an Amazon S3 bucket, rather than a local file path.
s3credentials	A list of length three, providing the credentials for accessing files in a private Amazon S3 bucket.
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 <code>native = TRUE</code> increases HDF5 file portability between programming languages. A file written with <code>native = TRUE</code> should also be read with <code>native = TRUE</code>

**Value**

h5ls returns a data.frame with the file content.

**Author(s)**

Bernd Fischer, Mike L. Smith

**References**

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

**See Also**

[h5dump\(\)](#)

**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")

# list content of an hdf5 file in a public S3 bucket

h5ls(file = "https://rhdf5-public.s3.eu-central-1.amazonaws.com/h5ex_t_array.h5", s3 = TRUE)
```

---

H5Oclose

*Close an HDF5 object*

---

**Description**

Close an HDF5 object

**Usage**

```
H5Oclose(h5obj)
```

**Arguments**

h5obj                    An object of class [H5IdComponent](#) representing an open HDF5 object.

**See Also**

[H5Oopen\(\)](#)

---

H5Ocopy	<i>Copies an HDF5 object</i>
---------	------------------------------

---

**Description**

Copies an HDF5 object

**Usage**

```
H5Ocopy(h5loc, name, h5loc_dest, name_dest, obj_cpy_pl = NULL, lcp1 = NULL)
```

**Arguments**

h5loc                    An object of class [H5IdComponent](#) representing an open HDF5 object where the source object should be copied from.

name                    Character vector of length 1, giving the name of the source object to be copied.

h5loc\_dest              An object of class [H5IdComponent](#) representing an open HDF5 object where the new copy should be created.

name\_dest              Character vector of length 1, giving the name of the new object to be created.

obj\_cpy\_pl, lcp1        [H5IdComponent](#) objects representing object copy and link creation property lists respectively. If left as NULL the default values for these will be used.

**Examples**

```
## Create a temporary copy of an example file check the contents
example_file <- system.file("testfiles", "h5ex_t_array.h5", package="rhdf5")
file.copy(example_file, tempdir())
h5_file <- file.path(tempdir(), "h5ex_t_array.h5")
h5ls(h5_file)

## open the example file and create a new, empty, file
fid1 <- H5Fopen( h5_file )
h5_file2 <- tempfile(fileext = ".h5")
fid2 <- H5Fcreate( h5_file2 )

## We can copy a dataset inside the same file
H5Ocopy(h5loc = fid1, name = "DS1", h5loc_dest = fid1, name_dest = "DS2")
## Or to a different file
H5Ocopy(h5loc = fid1, name = "DS1", h5loc_dest = fid2, name_dest = "DS1_copy")
```

```

## if we want to create a new group hierarchy we can use a link creation property list
lcpl <- H5Pcreate("H5P_LINK_CREATE")
H5Pset_create_intermediate_group( lcpl, create_groups = TRUE )
H5Ocopy(h5loc = fid1, name = "DS1", h5loc_dest = fid2, name_dest = "/foo/baa/DS1_nested", lcpl = lcpl)

## tidy up
H5Pclose(lcpl)
H5Fclose(fid1)
H5Fclose(fid2)

## Check we now have groups DS1 and DS2 in the original file
h5ls( h5_file )
## Check we have a copy of DS1 at the root and nests in the new file
h5ls( h5_file2 )

```

---

H5Oget\_num\_attrs

*Find the number of attributes associated with an HDF5 object*


---

## Description

Find the number of attributes associated with an HDF5 object

## Usage

```
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).
h5loc	An object of class <a href="#">H5IdComponent</a> representing a H5 location identifier (file or group).
name	The name of the object to be checked.

## Details

These functions are not part of the standard HDF5 C API.

## Value

Returns a vector of length 1 containing the number of attributes the specified object has.



---

**H5Olink***Create a hard link to an object in an HDF5 file*

---

**Description**

Create a hard link to an object in an HDF5 file

**Usage**

```
H5Olink(h5obj, h5loc, newLinkName, lcp1 = NULL, lap1 = NULL)
```

**Arguments**

<code>h5obj</code>	An object of class <a href="#">H5IdComponent</a> representing the object to be linked to.
<code>h5loc</code>	An object of class <a href="#">H5IdComponent</a> representing the location at which the object is to be linked. Can represent a file, group, dataset, datatype or attribute.
<code>newLinkName</code>	Character string giving the name of the new link. This should be relative to <code>h5loc</code> .
<code>lcp1, lap1</code>	<a href="#">H5IdComponent</a> objects representing link creation and link access property lists respectively. If left as NULL the default values for these will be used.

**See Also**

[H5Gcreate\\_anon](#)

**Examples**

```
## Create a temporary copy of an example file, and open it
example_file <- system.file("testfiles", "h5ex_t_array.h5", package="rhdf5")
file.copy(example_file, tempdir())
h5_file <- file.path(tempdir(), "h5ex_t_array.h5")
fid <- H5Fopen( h5_file )

## create a new group without a location in the file
gid <- H5Gcreate_anon(fid)

## create link to newly create group
## relative to the file identifier
H5Olink(h5obj = gid, h5loc = fid, newLinkName = "foo")

## tidy up
H5Gclose(gid)
H5Fclose(fid)

## Check we now have a "/foo" group
h5ls( h5_file )
```

---

H5Oopen

*Open an object in an HDF5 file*

---

### Description

Open an object in an HDF5 file

### Usage

```
H5Oopen(h5loc, name)
```

### Arguments

h5loc	An object of class <a href="#">H5IdComponent</a>
name	Path to the object to be opened. This should be relative to h5loc rather than the file.

### Value

An object of class [H5IdComponent](#) if the open operation was successful. FALSE otherwise.

### See Also

[H5Oclose\(\)](#)

### 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)
```

---

H5Pall\_filters\_avail *Query dataset filter properties.*

---

### Description

Return information about the filter pipeline applied to a dataset creation property list.

### Usage

```
H5Pall_filters_avail(h5plist)
```

```
H5Pget_nfilters(h5plist)
```

```
H5Pget_filter(h5plist, idx)
```

### Arguments

h5plist	Object of class <a href="#">H5IdComponent</a> representing a dataset creation property list.
idx	Integer of length 1. This argument selects which filter to return information about. Indexing is R-style 1-based.

### Details

- `H5Pall_filters_avail()` checks whether all filters required to process a dataset are available to **rhdf5**. This can be required if reading files created with other HDF5 software.
- `H5Pget_nfilters()` returns the number of filters in the dataset chunk processing pipeline.
- `H5Pget_filter()` provides details of a specific filter in the pipeline. This includes the filter name and the parameters provided to it e.g. compression level.

---

H5Pclose *Close and release a property list*

---

### Description

`H5Pclose()` terminates access to a property list. All property lists should be closed when they no longer need to be accessed. This frees resources used by the property list. Failing to call `H5Pclose()` can lead to memory leakage over time.

### Usage

```
H5Pclose(h5plist)
```

### Arguments

h5plist	<a href="#">H5IdComponent</a> object representing the property list to close.
---------	---

---

H5Pcopy	<i>Copy an existing property list to create a new property list</i>
---------	---

---

**Description**

Copy an existing property list to create a new property list

**Usage**

```
H5Pcopy(h5plist)
```

**Arguments**

h5plist      [H5IdComponent](#) object representing the property list to be copied.

---

H5Pcreate	<i>Create a new HDF5 property list</i>
-----------	--

---

**Description**

Create a new HDF5 property list

**Usage**

```
H5Pcreate(type = h5default("H5P"), native = FALSE)
```

**Arguments**

type	A character name of a property list type. See <code>h5const("H5P")</code> for possible property list types.
native	Defunct! Doesn't achieve anything for property lists.

---

H5Pfill\_value\_defined *Determine whether a property list has a fill value defined*

---

**Description**

Determine whether a property list has a fill value defined

**Usage**

```
H5Pfill_value_defined(h5plist)
```

**Arguments**

h5plist            Object of class [H5IdComponent](#) representing a dataset creation property list.

**Details**

Note that the return value for this function is slightly different from the C version. The C API provides three return types and can, in the case that a fill value is defined, differentiate whether the value is the HDF5 library default or has been set by the application.

**Value**

TRUE if the fill value is defined, FALSE if not. Will return NULL if there is a problem determining the status of the fill value.

---

H5Pget\_class            *Return the property list class identifier for a property list*

---

**Description**

Return the property list class identifier for a property list

**Usage**

```
H5Pget_class(h5plist)
```

**Arguments**

h5plist            [H5IdComponent](#) object representing any type of HDF5 property list.

---

H5Pget_version	<i>Get version information for objects in a file creation property list</i>
----------------	---

---

**Description**

Get version information for objects in a file creation property list

**Usage**

```
H5Pget_version(h5plist)
```

**Arguments**

h5plist      [H5IdComponent](#) object representing the file creation property list

**Value**

Named integer vector

---

H5Pobject_track_times	<i>Set whether to record timestamps for operations performed on an HDF5 object.</i>
-----------------------	---

---

**Description**

Set whether to record timestamps for operations performed on an HDF5 object.

**Usage**

```
H5Pset_obj_track_times(h5plist, track_times = TRUE)
```

```
H5Pget_obj_track_times(h5plist)
```

**Arguments**

h5plist      An [H5IdComponent](#) object representing an object creation property list.  
 track\_times      logical specifying whether times associated with an object should recorded.

**Details**

Objects created using high-level **rhdf5** functions like [h5createDataset\(\)](#) will have this setting turned off. This was done to ensure otherwise identical files returned the same md5 hash. This differs from the default setting in HDF5, which is for objects to record the times operations were performed on them.

---

H5Pset_blosc	<i>Add the BLOSC filter to the chunk processing pipeline.</i>
--------------	---

---

**Description**

Add the BLOSC filter to the chunk processing pipeline.

**Usage**

```
H5Pset_blosc(h5plist, h5tid, method = 1L, level = 6L, shuffle = TRUE)
```

**Arguments**

h5plist	Object of class <a href="#">H5IdComponent</a> representing a dataset creation property list.
h5tid	HDF5 data type id
method	Integer defining which of the compression algorithms provided by BLOSC should be used. (See the details section for the mapping between integers and algorithms).
level	Compression level to be used by the selected algorithm.
shuffle	Logical defining whether the bit-shuffle algorithm should be used prior to compression. This makes use of the shuffle implementation provide by BLOSC, rather than the HDF5 version.

---

H5Pset_bzip2	<i>Add the BZIP2 filter to the chunk processing pipeline.</i>
--------------	---

---

**Description**

Add the BZIP2 filter to the chunk processing pipeline.

**Usage**

```
H5Pset_bzip2(h5plist, level = 2L)
```

**Arguments**

h5plist	Object of class <a href="#">H5IdComponent</a> representing a dataset creation property list.
level	Compression level to be used by the selected algorithm.

---

H5Pset_deflate	<i>Add the deflate compression filter to the chunk processing pipeline.</i>
----------------	---

---

### Description

Valid values for the compression level range from 0 (no compression) to 9 (best compression, slowest speed). Note that applying this function with `level = 0` does not mean the filter is removed. It is still part of the filter pipeline, but no compression is performed. The filter will still need to be available on any system that reads a file created with this setting

### Usage

```
H5Pset_deflate(h5plist, level)
```

### Arguments

<code>h5plist</code>	Object of class <a href="#">H5IdComponent</a> representing a dataset creation property list.
<code>level</code>	Integer giving the compression level to use. Valid values are from 0 to 9.

---

H5Pset_fapl_ros3	<i>Set the read-only S3 virtual file driver</i>
------------------	---

---

### Description

The read-only S3 virtual file driver can be used to read files hosted remotely on Amazon's S3 storage.

### Usage

```
H5Pset_fapl_ros3(h5plist, s3credentials = NULL)
```

### Arguments

<code>h5plist</code>	<a href="#">H5IdComponent</a> object representing a file access property list.
<code>s3credentials</code>	Either NULL or a list of length 3 specifying the AWS access credentials (see details).

### Details

To access files in a private Amazon S3 bucket you will need to provide three additional details: The AWS region where the files are hosted, your AWS access key ID, and your AWS secret access key. More information on how to obtain AWS access keys can be found at <https://docs.aws.amazon.com/general/latest/gr/aws-sec-cred-types.html#access-keys-and-secret-access-keys>. These are provided as a list to the `s3credentials` argument. If you are accessing public data this argument should be NULL.



**Examples**

```
## this doesn't work on the Bioconductor Mac build machine
## Not run:
pid <- H5Pcreate("H5P_FILE_ACCESS")
H5Pset_fapl_ros3( pid )
H5Pclose(pid)

## End(Not run)
```

---

H5Pset\_istore\_k      *Get and set the 1/2 rank of an indexed storage B-tree*

---

**Description**

Get and set the 1/2 rank of an indexed storage B-tree

**Usage**

```
H5Pset_istore_k(h5plist, ik)

H5Pget_istore_k(h5plist)
```

**Arguments**

h5plist	<a href="#">H5IdComponent</a> object representing the file creation property list
ik	chunked Storage B-tree 1/2 rank

---

H5Pset\_lzf      *Add the LZF filter to the chunk processing pipeline.*

---

**Description**

Add the LZF filter to the chunk processing pipeline.

**Usage**

```
H5Pset_lzf(h5plist, h5tid)
```

**Arguments**

h5plist	Object of class <a href="#">H5IdComponent</a> representing a dataset creation property list.
h5tid	HDF5 data type id

---

H5Pset_nbit	<i>Add the N-Bit filter to the chunk processing pipeline.</i>
-------------	---

---

**Description**

Add the N-Bit filter to the chunk processing pipeline.

**Usage**

```
H5Pset_nbit(h5plist)
```

**Arguments**

h5plist	Object of class <a href="#">H5IdComponent</a> representing a dataset creation property list.
---------	--

**Value**

Returns (invisibly) an integer vector of length 1. The only element of this vector will be non-negative if the filter was set successfully and negative otherwise.

---

H5Pset_shared_mesg_index	<i>Get and set shared object header message index properties</i>
--------------------------	--

---

**Description**

Get and set shared object header message index properties

**Usage**

```
H5Pset_shared_mesg_index(  
    h5plist,  
    index_num,  
    mesg_type_flags = h5default(type = "H5O_SHMESG_FLAG"),  
    min_mesg_size  
)
```

```
H5Pget_shared_mesg_index(h5plist, index_num)
```

**Arguments**

h5plist	<a href="#">H5IdComponent</a> object representing the file creation property list
index_num	Index being configured. Indices use C-style 0-based counting, so the first index will be numbered 0.
mesg_type_flags	Character specifying the types of messages that may be stored in this index. Valid values can be found with <code>h5const(type = "H5O_SHMESG_FLAG")</code>
min_mesg_size	Minimum message size

**Value**

H5Pget\_shared\_mesg\_index() returns a list of length 2. The first element is the types of messages that may be stored in the index, the second element is the minimum message size.

---

H5Pset\_shared\_mesg\_nindexes

*Get and set the number of object header message indexes*

---

**Description**

Get and set the number of object header message indexes

**Usage**

H5Pset\_shared\_mesg\_nindexes(h5plist, nindexes)

H5Pget\_shared\_mesg\_nindexes(h5plist)

**Arguments**

h5plist	<a href="#">H5IdComponent</a> object representing the file creation property list
nindexes	Number of shared object header message indexes to be available in files

---

H5Pset\_shared\_mesg\_phase\_change

*Get and set threshold values for storage of shared object header message indexes*

---

**Description**

Get and set threshold values for storage of shared object header message indexes

**Usage**

H5Pset\_shared\_mesg\_phase\_change(h5plist, max\_list, min\_btree)

H5Pget\_shared\_mesg\_phase\_change(h5plist)

**Arguments**

h5plist	<a href="#">H5IdComponent</a> object representing the file creation property list
max_list	Threshold above which storage shifts from list to B-tree
min_btree	Threshold below which storage reverts to list format

---

H5Pset_shuffle	<i>Add the shuffle filter to the chunk processing pipeline.</i>
----------------	---

---

**Description**

Add the shuffle filter to the chunk processing pipeline.

**Usage**

```
H5Pset_shuffle(h5plist)
```

**Arguments**

h5plist	Object of class <a href="#">H5IdComponent</a> representing a dataset creation property list.
---------	--

**Value**

Returns (invisibly) an integer vector of length 1. The only element of this vector will be non-negative if the filter was set successfully and negative otherwise.

---

H5Pset_sizes	<i>Get and set the sizes of offsets and lengths used in an HDF5 file</i>
--------------	--

---

**Description**

Get and set the sizes of offsets and lengths used in an HDF5 file

**Usage**

```
H5Pset_sizes(h5plist, sizeof_addr, sizeof_size)
```

```
H5Pget_sizes(h5plist)
```

**Arguments**

h5plist	<a href="#">H5IdComponent</a> object representing the file creation property list
sizeof_addr	Offset size in bytes
sizeof_size	Length size in bytes

---

H5Pset_sym_k	<i>Get and set the size of the symbol table B-tree 1/2 rank and the leaf node 1/2 size</i>
--------------	--

---

**Description**

Get and set the size of the symbol table B-tree 1/2 rank and the leaf node 1/2 size

**Usage**

```
H5Pset_sym_k(h5plist, ik, lk)
```

```
H5Pget_sym_k(h5plist)
```

**Arguments**

h5plist	<a href="#">H5IdComponent</a> object representing the file creation property list
ik	Symbol table B-tree 1/2 rank
lk	Symbol table leaf node 1/2 size

---

H5Pset_szip	<i>Add the SZIP compression filter to the chunk processing pipeline.</i>
-------------	--

---

**Description**

Add the SZIP compression filter to the chunk processing pipeline.

**Usage**

```
H5Pset_szip(h5plist, options_mask, pixels_per_block)
```

**Arguments**

h5plist	Object of class <a href="#">H5IdComponent</a> representing a dataset creation property list.
options_mask, pixels_per_block	Integer vectors of length 1, setting parameters of the SZIP algorithm. See <a href="https://portal.hdfgroup.org/display/HDF5/H5P_SET_SZIP">https://portal.hdfgroup.org/display/HDF5/H5P_SET_SZIP</a> for more details.

**References**

<https://portal.hdfgroup.org/display/HDF5/Szip+Compression+in+HDF+Products>

---

H5Pset_userblock	<i>Get and set the user block size</i>
------------------	--

---

**Description**

Get and set the user block size

**Usage**

```
H5Pset_userblock(h5plist, size)
```

```
H5Pget_userblock(h5plist)
```

**Arguments**

h5plist	<a href="#">H5IdComponent</a> object representing the file creation property list
size	of the user block in bytes

---

H5P_chunk	<i>Get and set the size of the chunks used to store a chunked layout dataset</i>
-----------	--

---

**Description**

Get and set the size of the chunks used to store a chunked layout dataset

**Usage**

```
H5Pset_chunk(h5plist, dim)
```

```
H5Pget_chunk(h5plist)
```

**Arguments**

h5plist	An object of class <a href="#">H5IdComponent</a> representing a dataset creation property list.
dim	The chunk size used to store the dataset. This argument should be an integer vector of the same length as the number of dimensions of the dataset the dataset creation property list will be applied to.

**Details**

Note that a necessary side effect of running this function is that the layout of the dataset will be changes to H5D\_CHUNKED if it is not already set to this.

**See Also**

[H5Pset\\_layout\(\)](#)

---

H5P\_chunk\_cache      *Set parameters for the raw data chunk cache*

---

### Description

Set parameters for the raw data chunk cache

### Usage

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

### Arguments

h5plist	Object of class <a href="#">H5IdComponent</a> representing a dataset access property list.
rdcc_nslots	Integer defining the number of chunk slots in the raw data chunk cache for this dataset.
rdcc_nbytes	Integer setting the total size of the raw data chunk cache for this dataset in bytes. In most cases increasing this number will improve performance, as long as you have enough free memory. The default size is 1 MB
rdcc_w0	Numeric value defining the chunk preemption policy. Must be between 0 and 1 inclusive.

---

H5P\_create\_intermediate\_group  
*Get and set whether to create missing intermediate groups*

---

### Description

Get and set whether to create missing intermediate groups

### Usage

```
H5Pset_create_intermediate_group(h5plist, create_groups = TRUE)
```

```
H5Pget_create_intermediate_group(h5plist)
```

### Arguments

h5plist	An object of class <a href="#">H5IdComponent</a> representing a link creation property list.
create_groups	A logical of length 1 specifying whether missing groups should be created when a new object is created. Default is TRUE.

**Examples**

```
pid <- H5Pcreate("H5P_LINK_CREATE")

## by default intermediate groups are not created
H5Pget_create_intermediate_group( pid )

## Change the setting so groups will be created

H5Pget_create_intermediate_group( pid )

## tidy up
H5Pclose(pid)
```

---

H5P_fill_time	<i>Set the time when fill values are written to a dataset</i>
---------------	---

---

**Description**

Set the time when fill values are written to a dataset

**Usage**

```
H5Pset_fill_time(h5plist, fill_time = h5default("H5D_FILL_TIME"))

H5Pget_fill_time(h5plist)
```

**Arguments**

h5plist	An object of class <a href="#">H5IdComponent</a> representing a dataset creation property list.
fill_time	When the fill values should be written. Possible options can be listed with <code>h5const("H5D_FILL_TIME")</code> .

---

H5P_fill_value	<i>Set the fill value for an HDF5 dataset</i>
----------------	---

---

**Description**

H5Pset\_fill\_value sets the fill value for a dataset in the dataset creation property list.

**Usage**

```
H5Pset_fill_value(h5plist, value)
```



**Arguments**

h5plist	An object of class <a href="#">H5IdComponent</a> representing a dataset creation property list.
value	The default fill value of the dataset. A vector of length 1.

**See Also**

[H5P\\_fill\\_time](#), [H5Pfill\\_value\\_defined](#)

---

H5P_layout	<i>Get and set the type of storage used to store the raw data for a dataset</i>
------------	---

---

**Description**

Possible options for the layout argument are:

- H5D\_COMPACT
- H5D\_CONTIGUOUS
- H5D\_CHUNKED
- H5D\_VIRTUAL

**Usage**

```
H5Pset_layout(h5plist, layout = h5default("H5D"))
```

```
H5Pget_layout(h5plist)
```

**Arguments**

h5plist	An object of class <a href="#">H5IdComponent</a> representing a dataset creation property list.
layout	A character giving the name of a dataset layout type.

**Details**

The names of the layout types can also be obtained via `h5const("H5D")`.

---

H5P_libver_bounds	<i>Control the range of HDF5 library versions that will be compatible with a file.</i>
-------------------	--

---

### Description

Control the range of HDF5 library versions that will be compatible with a file.

### Usage

```
H5Pset_libver_bounds(  
  h5plist,  
  libver_low = "H5F_LIBVER_EARLIEST",  
  libver_high = "H5F_LIBVER_LATEST"  
)  
  
H5Pget_libver_bounds(h5plist)
```

### Arguments

h5plist            [H5IdComponent](#) object representing a file access property list.  
libver\_low, libver\_high            Define the earliest and latest versions of the HDF5 library that will be used when writing object in the file.

### Description

The H5R functions can be used for creating or working with references to specific objects and data regions in an HDF5 file.

### Author(s)

Mike Smith

### Examples

```
library(rhdf5)  
  
## first we'll create a file with a group named "foo" and a  
## 1-dimensional dataset named "baa" inside that group.  
file_name <- tempfile(fileext = ".h5")  
h5createFile(file_name)  
h5createGroup(file = file_name, group = "/foo")
```

```

h5write(1:100, file=file_name, name="/foo/baa")

fid <- H5Fopen(file_name)
ref_to_group <- H5Rcreate(fid, name = "/foo")
ref_to_dataset <- H5Rcreate(fid, name = "/foo/baa")
two_refs <- c(ref_to_group, ref_to_dataset)
two_refs

## the size of this dataspace is the number of object references
## we want to store
sid <- H5Screate_simple(2)
tid <- H5Tcopy(dtype_id = "H5T_STD_REF_OBJ")
did <- H5Dcreate(fid, name = "object_refs", dtype_id = tid, h5space = sid)
H5Dwrite(did, two_refs)
H5Dclose(did)
H5Sclose(sid)
H5Fclose(fid)

```

---

H5Rcreate

*Create a reference*


---

### Description

Creates a reference to an object or dataset selection inside an HDF5 file.

### Usage

```
H5Rcreate(h5loc, name, ref_type = "H5R_OBJECT", h5space = NULL)
```

### Arguments

h5loc	An H5IdComponent object representing the location to be pointed to by the created reference.
name	Character string giving the name of the object to be referenced, relative to the location given by h5loc.
ref_type	The type of reference to create. Accepts either H5R_OBJECT or H5R_DATASET_REGION.
h5space	An object of class H5IdComponent representing a dataspace with a selection set. This argument is only used if creating a reference to a dataset region, and will be ignored otherwise.

### Value

An [H5Ref](#) object storing the reference.

---

H5Rdereference	<i>Open a reference object.</i>
----------------	---------------------------------

---

**Description**

Given a reference and the file to which that reference applies, H5Rdereference() will open the reference object and return an identifier.

**Usage**

```
H5Rdereference(ref, h5loc)
```

**Arguments**

ref	H5ref object containing the reference to be opened.
h5loc	An H5IdComponent object representing the file containing the referenced object.

**Details**

If ref contains more than one reference, only the first reference will be used. It must be subset with [ if one of the other stored references should be opened.

**Value**

An object of class H5IdComponent representing the opened object referenced by ref. This should be closed with the appropriate function e.g. [H5Dclose\(\)](#), [H5Oclose\(\)](#), etc. when no longer needed.

---

H5Ref-class	<i>An S4 class representing H5 references.</i>
-------------	--

---

**Description**

A class representing one or more HDF5 references.

**Usage**

```
## S4 method for signature 'H5Ref'
show(object)

## S4 method for signature 'H5Ref'
length(x)

## S4 method for signature 'H5Ref'
c(x, ...)

## S4 method for signature 'H5Ref'
x[i]
```

**Arguments**

object	Object of class H5Ref
x	An H5Ref object.
...	Additional H5Ref objects to be combined with x.
i	Integer vector giving the indices of references to select.

**Details**

The length of the `val` slot is dependent on both the number and type of references stored in the object. `H5R_OBJECT` references are stored in 8 bytes, while `H5R_DATASET_REGION` references require 12 bytes. The length of `val` will then be a multiple of 8 or 12 respectively. This also means that references of different types cannot be combined in a single object.

**Methods (by generic)**

- `show(H5Ref)`: Print details of the object to screen.
- `length(H5Ref)`: Return the number of references stored in an H5Ref object.
- `c(H5Ref)`: Combine two or more H5Ref objects. Objects must all contain the same type of reference, either `H5R_OBJECT` or `H5R_DATASET_REFERENCE`.
- `[]`: Subset an H5Ref object.

**Slots**

`val` raw vector containing the byte-level representation of each reference.

`type` integer of length 1, which maps to either `H5R_OBJECT` or `H5R_DATASET_REGION`.

---

H5Rget\_name

*Return the name of the object that a reference points to*

---

**Description**

Return the name of the object that a reference points to

**Usage**

```
H5Rget_name(ref, h5loc)
```

**Arguments**

ref	H5ref object containing the reference to be queried.
h5loc	An H5IdComponent object representing the file containing the referenced object.

**Value**

Character string of length 1 giving the name of the referenced object.

---

H5Rget_obj_type	<i>Identify the type of object that a reference points to</i>
-----------------	---

---

**Description**

Identify the type of object that a reference points to

**Usage**

```
H5Rget_obj_type(ref, h5loc)
```

**Arguments**

ref	H5ref object containing the reference to be queried.
h5loc	An H5IdComponent object representing the file containing the referenced object.

**Value**

Character string of length 1 identifying the object type. Valid return values are: "GROUP", "DATASET", and "NAMED\_DATATYPE".

---

H5Rget_region	<i>Return selection for a reference to dataset region</i>
---------------	---

---

**Description**

Given a dataset region reference, this function will return the dataspace and selection required to read the data points indicated by the reference.

**Usage**

```
H5Rget_region(ref, h5loc)
```

**Arguments**

ref	An object of class H5Ref. This function is only valid for reference of type H5R_DATASET_REGION, and not H5R_OBJECT.
h5loc	An H5IdComponent object representing the file containing the referenced object.

**Value**

An object of class H5IdComponent representing the dataspace of the dataset that ref points to. The dataspace will have the selection set that matches the selection pointed to by ref. This should be closed using [H5Sclose\(\)](#) when no longer required.

---

H5Sclose	<i>Close and release a dataspace</i>
----------	--------------------------------------

---

**Description**

Close and release a dataspace

**Usage**

```
H5Sclose(h5space)
```

**Arguments**

h5space            Object of class [H5IdComponent](#) representing the dataspace to be closed.

**See Also**

[H5Screate\(\)](#)

---

H5Scombine_hyperslab	<i>Perform operation between an existing selection and an another hyperslab definition.</i>
----------------------	---

---

**Description**

Combines a hyperslab selection specified by start, stride, count and block arguments with the current selection for the dataspace represented by h5space.

**Usage**

```
H5Scombine_hyperslab(
    h5space,
    op = h5default("H5S_SELECT"),
    start = NULL,
    stride = NULL,
    count = NULL,
    block = NULL
)
```

**Arguments**

h5space            [H5IdComponent](#) object representing a dataspace.

op                 Character string defined the operation used to join the two dataspace. See [h5const\("H5S\\_SELECT"\)](#) for the list of available options.

start, stride, count, block            Integer vectors, each with length equal to the rank of the dataspace. These parameters define the new hyperslab to select.

**Value**

An [H5IdComponent](#) object representing a new dataspace with the generated selection.

**See Also**

[H5Scombine\\_select\(\)](#), [H5Sselect\\_hyperslab\(\)](#)

**Examples**

```
## create a 1 dimensional dataspace
sid_1 <- H5Screate_simple(dims = 20)

## select a single block of 5 points in sid_1
## this is equivalent to [11:16] in R syntax
H5Sselect_hyperslab(sid_1, start = 11, stride = 1,
                    block = 5, count = 1)#

## combine the existing selection with a new
## selection consisting of 2 blocks each of 1 point
## equivalent to [c(3,5)] in R syntax
sid_2 <- H5Scombine_hyperslab(sid_1, op = "H5S_SELECT_OR",
                              start = 3, stride = 2,
                              block = 1, count = 2)

## confirm we have selected 5 in our original dataspace
## and 7 points in the newly created dataspace
H5Sget_select_npoints(sid_1)
H5Sget_select_npoints(sid_2)

## tidy up
H5Sclose(sid_1)
H5Sclose(sid_2)
```

---

H5Scombine_select	<i>Combine two selections</i>
-------------------	-------------------------------

---

**Description**

Combine two selections

**Usage**

```
H5Scombine_select(h5space1, op = h5default("H5S_SELECT"), h5space2)
```

**Arguments**

h5space1, h5space2

[H5IdComponent](#) objects representing a dataspace.

op  
Character string defined the operation used to join the two dataspace. See [h5const\("H5S\\_SELECT"\)](#) for the list of available options.



**Value**

Returns an [H5IdComponent](#) object representing a new dataspace. The new dataspace will have the same extent as h5space1 with the hyperslab selection being the result of combining the selections of h5space1 and h5space2.

**See Also**

[H5Scombine\\_hyperslab\(\)](#)

**Examples**

```
## create two 1 dimensional dataspace
## of different sizes
sid_1 <- H5Screate_simple(dims = 20)
sid_2 <- H5Screate_simple(dims = 10)

## select a single block of 5 points in sid_1
## this is equivalent to [11:16] in R syntax
H5Sselect_hyperslab(sid_1, start = 11, stride = 1,
                    block = 5, count = 1)

## select 2 blocks of 1 point from sid_2
## equivalent to [c(3,5)] in R syntax
H5Sselect_hyperslab(sid_2, start = 3, stride = 2,
                    block = 1, count = 2)

## confirm we have select 5 and 2 points respectively
H5Sget_select_npoints(sid_1)
H5Sget_select_npoints(sid_2)

## combine the two dataset selections keeping points that
## are in one or both of the selections
sid_3 <- H5Scombine_select(sid_1, "H5S_SELECT_OR", sid_2)

## extent of the new dataset is the same as sid_1
sid_3
## confirm the selection contains 7 points
H5Sget_select_npoints(sid_3)

## tidy up
H5Sclose(sid_1)
H5Sclose(sid_2)
H5Sclose(sid_3)
```

**Description**

H5S\_copy() creates an exact copy of a given dataspace.

**Usage**

```
H5Scopy(h5space)
```

**Arguments**

h5space            Object of class [H5IdComponent](#) representing the dataspace to be copied.

**Value**

If the copying is successful returns an object of class [H5IdComponent](#) representing the new dataspace. Otherwise returns FALSE.

---

H5Screate	<i>Create a new dataspace of a specified type</i>
-----------	---

---

**Description**

Create a new dataspace of a specified type

**Usage**

```
H5Screate(type = h5default("H5S"), native = FALSE)
```

**Arguments**

type            The type of dataspace to create. See `h5const("H5S")` for possible types.

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 an object of class [H5IdComponent](#) representing a dataspace.

**See Also**

[H5Screate\\_simple](#)

---

H5Screate\_simple      *Create a simple dataspace*

---

**Description**

Create a simple dataspace

**Usage**

```
H5Screate_simple(dims, maxdims, native = FALSE)
```

**Arguments**

dims	An integer vector defining the initial dimensions of the dataspace. The length of dims determines the rank of the dataspace.
maxdims	An integer vector with the same length length as dims. Specifies the upper limit on the size of the dataspace dimensions. Only needs to be specified if this is different from the values given 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.

**Value**

Returns an object of class [H5IdComponent](#) representing a dataspace.

**See Also**

[H5Screate](#)

---

H5Sget\_select\_npoints      *Find the number of elements in a dataspace selection*

---

**Description**

Find the number of elements in a dataspace selection

**Usage**

```
H5Sget_select_npoints(h5space)
```

**Arguments**

h5space	<a href="#">H5IdComponent</a> object representing a dataspace.
---------	--

---

H5Sget\_simple\_extent\_dims  
*Find the size of a dataspace*

---

**Description**

Find the size of a dataspace

**Usage**

H5Sget\_simple\_extent\_dims(h5space)

**Arguments**

h5space      [H5IdComponent](#) object representing a dataspace.

---

H5Sis\_simple      *Determine whether a dataspace is a simple dataspace*

---

**Description**

In HDF5 a dataspace is considered "simple" if it represents a regular N-dimensional array of points. Currently (HDF 1.10.7) all dataspaces are simple. Support for complex dataspaces is planned for future HDF versions.

**Usage**

H5Sis\_simple(h5space)

**Arguments**

h5space      [H5IdComponent](#) object representing a dataspace.

---

H5Sselect\_all      *Set the selection region of a dataspace to include all elements*

---

**Description**

Set the selection region of a dataspace to include all elements

**Usage**

H5Sselect\_all(h5space)

**Arguments**

h5space      [H5IdComponent](#) object representing a dataspace.

---

H5Sselect\_hyperslab     *Perform operation between an existing selection and an another hyperslab definition.*

---

### Description

Combines a hyperslab selection specified by start, stride, count and block arguments with the current selection for the dataspace represented by h5space.

### Usage

```
H5Sselect_hyperslab(
  h5space,
  op = h5default("H5S_SELECT"),
  start = NULL,
  stride = NULL,
  count = NULL,
  block = NULL
)
```

### Arguments

h5space     [H5IdComponent](#) object representing a dataspace.

op     Character string defined the operation used to join the two dataspace. See `h5const("H5S_SELECT")` for the list of available options.

start, stride, count, block     Integer vectors, each with length equal to the rank of the dataspace. These parameters define the new hyperslab to select.

### Details

H5Sselect\_hyperslab is similar to, but subtly different from, [H5Scombine\\_hyperslab\(\)](#). The former modifies the selection of the dataspace provided in the h5space argument, while the later returns a new dataspace with the combined selection.

### Examples

```
## create a 1 dimensional dataspace
sid_1 <- H5Screate_simple(dims = 20)

## select a single block of 5 points in sid_1
## this is equivalent to [11:16] in R syntax
H5Sselect_hyperslab(sid_1, start = 11, stride = 1,
  block = 5, count = 1)

## confirm we have selected 5 in our original dataspace
H5Sget_select_npoints(sid_1)
```

```
## combine the existing selection with a new
## selection consisting of 2 blocks each of 1 point
## equivalent to [c(3,5)] in R syntax
H5Sselect_hyperslab(sid_1, op = "H5S_SELECT_OR",
                    start = 3, stride = 2,
                    block = 1, count = 2)

## The dataspace now has 7 points selected
H5Sget_select_npoints(sid_1)

## tidy up
H5Sclose(sid_1)
```

---

H5Sselect\_index

*Select elements of a dataspace using R-style indexing*


---

### Description

Combines a hyperslab selection specified by `start`, `stride`, `count` and `block` arguments with the current selection for the dataspace represented by `h5space`.

### Usage

```
H5Sselect_index(h5space, index)
```

### Arguments

<code>h5space</code>	<a href="#">H5IdComponent</a> object representing a dataspace.
<code>index</code>	A list of integer indices. The length of the list corresponds to the number of dimensions of the HDF5 array. If a list element is <code>NULL</code> , all elements of the respective dimension are selected.

### Details

`H5Sselect_hyperslab` is similar to, but subtly different from, [H5Scombine\\_hyperslab\(\)](#). The former modifies the selection of the dataspace provided in the `h5space` argument, while the later returns a new dataspace with the combined selection.

### Examples

```
## create a 1 dimensional dataspace
sid <- H5Screate_simple(c(10,5,3))

## Select elements that lie in in the rows 1-3, columns 2-4,
## and the entire 3rd dimension
H5Sselect_index(sid, list(1:3, 2:4, NULL))

## We can check the number of selected points.
```

```
## This should be 27 (3 * 3 * 3)
H5Sget_select_npoints(sid)

## always close dataspace after usage to free resources
H5Sclose(sid)
```

---

H5Sselect_none	<i>Set the selection region of a dataspace to include no elements</i>
----------------	---

---

**Description**

Set the selection region of a dataspace to include no elements

**Usage**

```
H5Sselect_none(h5space)
```

**Arguments**

h5space      [H5IdComponent](#) object representing a dataspace.

---

H5Sselect_valid	<i>Check that a selection is valid</i>
-----------------	--

---

**Description**

Check that a selection is valid

**Usage**

```
H5Sselect_valid(h5space)
```

**Arguments**

h5space      [H5IdComponent](#) object representing a dataspace.

---

H5Sset\_extent\_simple *Set the size of a dataspace*

---

### Description

Set the size of a dataspace

### Usage

```
H5Sset_extent_simple(h5space, dims, maxdims)
```

### Arguments

h5space	<a href="#">H5IdComponent</a> object representing a dataspace.
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.

---

H5Sunlimited *Retrieve value for H5S\_UNLIMITED constant*

---

### Description

The value for H5S\_UNLIMITED can be provided to the maxdims argument of [H5Screate\\_simple](#) to indicate that the maximum size of the corresponding dimension is unlimited.

### Usage

```
H5Sunlimited()
```

### See Also

[H5Screate\\_simple](#)



---

H5Tcopy	<i>Copy an existing datatype</i>
---------	----------------------------------

---

**Description**

Copy an existing datatype

**Usage**

```
H5Tcopy(dtype_id = h5default(type = "H5T"))
```

**Arguments**

dtype_id	Datatype to copy. Can either be a character specifying a predefined HDF5 datatype (see <code>h5const("H5T")</code> for valid options) or the ID of an already created datatype.
----------	---

---

H5Tis_variable_str	<i>Determine whether a datatype is a variable length string</i>
--------------------	---

---

**Description**

Determine whether a datatype is a variable length string

**Usage**

```
H5Tis_variable_str(dtype_id)
```

**Arguments**

dtype_id	ID of HDF5 datatype to query.
----------	-------------------------------

---

H5T_cset	<i>Retrieve or set the character set to be used in a string datatype.</i>
----------	---

---

**Description**

Retrieve or set the character set to be used in a string datatype.

**Usage**

```
H5Tset_cset(dtype_id, cset = "ASCII")
```

```
H5Tget_cset(dtype_id)
```

**Arguments**

dtype_id	ID of HDF5 datatype to query or modify.
cset	Encoding to use for string types. Valid options are 'ASCII' and 'UTF-8'.

---

H5T_precision	<i>Retrieve or set the precision of an HDF5 datatype</i>
---------------	--

---

**Description**

Retrieve or set the precision of an HDF5 datatype

**Usage**

```
H5Tset_precision(dtype_id, precision)
```

```
H5Tget_precision(dtype_id)
```

**Arguments**

dtype_id	ID of HDF5 datatype to set precision of.
precision	The number of bytes of precision for the datatype.

**Value**

- H5Tget\_precision() returns an integer giving the number of significant bits used by the given datatype.
- H5Tset\_precision() is call for its side-effect of modifying the precision of a datatype. It will invisibly return TRUE if this is successful and will stop with an error if the operation fails.

---

H5T_size	<i>Retrieve or set the type of padding used by string datatype</i>
----------	--

---

**Description**

Retrieve or set the type of padding used by string datatype

**Usage**

```
H5Tset_size(dtype_id = h5default(type = "H5T"), size)
```

```
H5Tget_size(dtype_id)
```

**Arguments**

dtype_id	ID of HDF5 datatype to query or modify.
size	The new datatype size in bytes.

---

H5T_strpad	<i>Retrieve or set the type of padding used by string datatype</i>
------------	--

---

**Description**

Retrieve or set the type of padding used by string datatype

**Usage**

```
H5Tset_strpad(dtype_id, strpad = "NULLPAD")
```

```
H5Tget_strpad(dtype_id)
```

**Arguments**

dtype_id	ID of HDF5 datatype to query or modify.
strpad	Character vector of length 1 specifying the type of padding to use. Valid options are NULLTERM, NULLPAD and SPACEPAD.

---

h5version	<i>Print the rhdf5 and libhdf5 version numbers</i>
-----------	--

---

**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, Mike L. Smith

**Examples**

```
h5version()
```

---

H5Zfilter_avail	<i>Determine whether a filter is available on this system</i>
-----------------	---

---

**Description**

Determine whether a filter is available on this system

**Usage**

```
H5Zfilter_avail(filter_id)
```

**Arguments**

`filter_id` Integer representing the ID of the filter to be checked.

---

h5\_createAttribute      *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,
  encoding = NULL,
  cset = NULL,
  native = FALSE
)
```

### 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.
attr	Name of the attribute to be created.
dims	The dimensions of the attribute as a numeric vector. If NULL, a scalar dataspace will be created instead.
maxdims	The maximum extension of the 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 <a href="#">H5IdComponent</a> representing 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> .
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	The maximum string length when storage.mode='character'. If this is specified, HDF5 stores each string of attr as fixed length character arrays. Together with compression, this should be efficient. If this argument is set to NULL, HDF5 will instead store variable-length strings.
encoding	The encoding of the string data type i.e. when storage.mode = 'character'. Valid options are "ASCII" and "UTF-8".
cset	<i>Deprecated in favour of the encoding argument.</i>
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)
```

---

h5_createDataset	<i>Create HDF5 dataset</i>
------------------	----------------------------

---

### 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,
  encoding = NULL,
  chunk = dims,
  fillValue,
  level = 6,
  filter = "gzip",
  shuffle = TRUE,
  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 <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. 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 <a href="#">H5Sunlimited()</a> to indicate an extensible dimension.
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 to use. The default value of NULL will result in using variable length strings. See the details for more information on this option.
encoding	The encoding of the string data type. Valid options are "ASCII" or "UTF-8".
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).
fillValue	Standard value for filling the dataset. The storage.mode of value has to be convertible to the dataset type by HDF5.
level	The compression level used. An integer value between 0 (no compression) and 9 (highest and slowest compression).
filter	Character defining which compression filter should be applied to the chunks of the dataset. See the Details section for more information on the options that can be provided here.
shuffle	Logical defining whether the byte-shuffle algorithm should be applied to data prior to compression.
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.

The size argument is only used when storage.mode = 'character'. When storing strings HDF5 can use either a fixed or variable length datatype. Setting size to a positive integer will use fixed length strings where size defines the length. **rhdf5** writes null padded strings by default and so to avoid data loss the value provided here should be the length of the longest string. Setting size = NULL will use variable length strings. The choice is probably dependent on the nature of the strings you're writing. The principle difference is that a dataset of variable length strings will not be compressed by HDF5 but each individual string only uses the space it requires, whereas in a fixed length dataset each string is of length uses size, but the whole dataset can be compressed. This explored more in the examples below.

The filter argument can take several options matching to compression filters distributed in either with the HDF5 library in **Rhdf5lib** or via the **rhdf5filters** package. The plugins available and the corresponding values for selecting them are shown below:

**zlib: Ubiquitous deflate compression algorithm used in GZIP or ZIP files. All three options below achieve the same r**

- "GZIP",
- "ZLIB",
- "DEFLATE"

**szip: Compression algorithm maintained by the HDF5 group.** • "SZIP"

**bzip2** • "BZIP2"



**BLOSC meta compressor:** As a meta-compressor BLOSC wraps several different compression algorithms. Each of t

"BLOSC\_BLOSC LZ"

- "BLOSC\_LZ4"
- "BLOSC\_LZ4HC"
- "BLOSC\_SNAPPY"
- "BLOSC\_ZLIB"
- "BLOSC\_ZSTD"

**lzf** • "LZF"

**Disable:** It is possible to write chunks without any compression applied. • "NONE"

### Value

Returns (invisibly) TRUE if dataset was created successfully and FALSE otherwise.

### Author(s)

Bernd Fischer, Mike L. Smith

### See Also

[h5createFile\(\)](#), [h5createGroup\(\)](#), [h5read\(\)](#), [h5write\(\)](#)

### 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 dataset with bzip2 compression
h5createDataset("ex_createDataset.h5", "D", c(5,8), storage.mode = "integer",
  chunk=c(5,1), filter = "BZIP2", level=6)

# create a dataset of strings & define size based on longest string
ex_strings <- c('long', 'longer', 'longest')
h5createDataset("ex_createDataset.h5", "E",
  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 = "E")
```

```
h5dump("ex_createDataset.h5")

## Investigating fixed vs variable length string datasets

## create 1000 random strings with length between 50 and 100 characters
words <- ceiling(runif(n = 1000, min = 50, max = 100)) |>
vapply(FUN = \(x) {
  paste(sample(letters, size = x, replace = TRUE), collapse = "")
},
FUN.VALUE = character(1))

## create two HDF5 files
f1 <- tempfile()
f2 <- tempfile()
h5createFile(f1)
h5createFile(f2)

## create two string datasets
## the first is variable length strings, the second fixed at the length of our longest word
h5createDataset(f1, "strings", dims = length(words), storage.mode = "character", size = NULL, chunk = 25)
h5createDataset(f2, "strings", dims = length(words), storage.mode = "character", size = max(nchar(words)), chunk = 25)

## Write the data
h5write(words, f1, "strings")
h5write(words, f2, "strings")

## Check file sizes.
## In this example the fixed length string dataset is normally much smaller
file.size(f1)
file.size(f2)
```

---

h5\_createFile

*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 (invisibly) TRUE if file was created successfully and FALSE otherwise.

**Author(s)**

Bernd Fischer

**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")
```

---

h5_createGroup	<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 <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.
group	The name of the new group. The name can contain a hierarchy of groupnames, e.g. <code>"/group1/group2/newgroup"</code> , but the function will fail if the top level groups do not exist.

**Details**

Creates a new group within an HDF5 file.

**Value**

Returns TRUE if group was created successfully and FALSE otherwise.

**Author(s)**

Bernd Fischer

**See Also**

[h5createFile\(\)](#), [h5createDataset\(\)](#), [h5read\(\)](#), [h5write\(\)](#)

**Examples**

```
h5createFile("ex_createGroup.h5")

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

h5ls("ex_createGroup.h5")
```

---

h5\_delete

*Delete objects within a HDF5 file*

---

**Description**

Deletes the specified 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	For h5delete the name of the object to be deleted. For h5deleteAttribute the name of the object to which the attribute belongs.

**Author(s)**

Mike Smith

---

h5_deleteAttribute	<i>Delete attribute</i>
--------------------	-------------------------

---

**Description**

Deletes an attribute associated with a group or dataset within an HDF5 file.

**Usage**

```
h5deleteAttribute(file, name, attribute)
```

**Arguments**

file	The filename (character) of the file in which the object is located.
name	The name of the object to which the attribute belongs.
attribute	Name of the attribute to be deleted.

**Author(s)**

Mike Smith

---

h5_dump	<i>Dump the content of an HDF5 file.</i>
---------	--

---

**Description**

Dump the content of an HDF5 file.

**Usage**

```
h5dump(  
  file,  
  recursive = TRUE,  
  load = TRUE,  
  all = FALSE,  
  index_type = h5default("H5_INDEX"),  
  order = h5default("H5_ITER"),  
  s3 = FALSE,  
  s3credentials = NULL,  
  ...,  
  native = FALSE  
)
```

**Arguments**

file	The filename (character) of the file in which the dataset will be located. You can also 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.
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.
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.
all	If TRUE, a longer list of information on each entry is provided.
index_type	See <code>h5const("H5_INDEX")</code> for possible arguments.
order	See <code>h5const("H5_ITER")</code> for possible arguments.
s3	Logical value indicating whether the file argument should be treated as a URL to an Amazon S3 bucket, rather than a local file path.
s3credentials	A list of length three, providing the credentials for accessing files in a private Amazon S3 bucket.
...	Arguments passed to <a href="#">h5read()</a>
native	An object of class <code>logical</code> . If TRUE, array-like objects are treated as stored in HDF5 row-major rather than R column-major orientation. Using <code>native = TRUE</code> increases HDF5 file portability between programming languages. A file written with <code>native = TRUE</code> should also be read with <code>native = TRUE</code>

**Value**

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, Mike L. Smith

**See Also**

[h5ls\(\)](#)

**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
h5dump("ex_ls_dump.h5")

# list content of an hdf5 file in a public S3 bucket

h5dump(file = "https://rhdf5-public.s3.eu-central-1.amazonaws.com/h5ex_t_array.h5", s3 = TRUE)
```

---

h5\_errorHandling      *Set how HDF5 error messages are displayed*

---

### Description

Sets the options for handling HDF5 error messages in the R sessions.

### 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.

### Value

Returns 0 if options are set successfully.

### Author(s)

Bernd Fischer

### See Also

[rhdf5](#)

### Examples

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

---

`h5_FileLocking`*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

<code>location</code>	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-existent 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 way 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` environment variable.

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()
}
```

---

h5\_read

*Reads and write object in HDF5 files*

---

**Description**

Reads objects in HDF5 files. This function can be used to read either full arrays/vectors or subarrays (hyperslabs) from 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,
  s3 = FALSE,
  s3credentials = NULL
)
```

**Arguments**

file	The filename (character) of the file in which the dataset is be located. 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.
name	The name of the dataset in the HDF5 file.
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> 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 <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 index 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 index is not <code>NULL</code> .
count	The number of blocks to be read. This argument is ignored, if index is not <code>NULL</code> .
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.
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.
read.attributes	(logical) If <code>TRUE</code> , the HDF5 attributes are read and attached to the respective R object.
drop	(logical) If <code>TRUE</code> , the HDF5 object is read as a vector with <code>NULL</code> dim attributes.
...	Further arguments passed to <code>H5Dread</code> .
native	An object of class <code>logical</code> . If <code>TRUE</code> , array-like objects are treated as stored in HDF5 row-major rather than R column-major orientation. Using <code>native = TRUE</code> increases HDF5 file portability between programming languages. A file written with <code>native = TRUE</code> should also be read with <code>native = TRUE</code>
s3	Logical value indicating whether the file argument should be treated as a URL to an Amazon S3 bucket, rather than a local file path.
s3credentials	A list of length three, providing the credentials for accessing files in a private Amazon S3 bucket.

## Details

Read an R object from an HDF5 file. If none of the arguments `start`, `stride`, `block`, `count` are 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 subarrays, so called hyperslabs from 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.

## Author(s)

Bernd Fischer, Mike Smith

## See Also

[h5ls](#)

## Examples

```
h5createFile("ex_hdf5file.h5")

# write a matrix
B = array(seq(0.1,2.0,by=0.1),dim=c(5,2,2))
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))

# Read a subset of an hdf5 file in a public S3 bucket

h5read('https://rhdf5-public.s3.eu-central-1.amazonaws.com/rhdf5ex_t_float_3d.h5',
      s3 = TRUE, name = "a1", index = list(NULL, 3, NULL))
```

---

h5_readAttributes	<i>Read all attributes from a given location in an HDF5 file</i>
-------------------	--

---

**Description**

Read all attributes from a given location in an HDF5 file

**Usage**

```
h5readAttributes(file, name, native = FALSE, ...)
```

**Arguments**

file	Character vector of length 1, giving the path to the HDF5
name	Path within the HDF5 file to the object whose attributes should be read.
native	An object of class <code>logical</code> . If <code>TRUE</code> , array-like objects are treated as stored in HDF5 row-major rather than R column-major orientation.
...	Further arguments passed to <a href="#">H5Aread</a> .

**Value**

A named list of the same length as the number of attributes attached to the specific object. The names of the list entries correspond to the attribute names. If no attributes are found an empty list is returned.

---

h5_save	<i>Saves a one or more objects to an HDF5 file.</i>
---------	---

---

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

### 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

### See Also

`h5ls()`, `h5write()`

### 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")
```

---

h5\_set\_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 <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	The name of the dataset in the HDF5 file, or an object of class <a href="#">H5IdComponent</a> representing a H5 dataset identifier. See <a href="#">H5Dcreate</a> , or <a href="#">H5Dopen</a> 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 program (e.g. <code>HD-FView</code> ), 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 <code>logical</code> . If <code>TRUE</code> , array-like objects are treated as stored in HDF5 row-major rather than R column-major orientation. Using <code>native = TRUE</code> increases HDF5 file portability between programming languages. A file written with <code>native = TRUE</code> should also be read with <code>native = TRUE</code>

**Value**

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

**Author(s)**

Bernd Fischer

**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")]
```

---

h5\_write

*Write object to an HDF5 file.*

---

**Description**

Writes an R object to an HDF5 file. This function can be used to write either full arrays/vectors or subarrays (hyperslabs) within an existing dataset.

**Usage**

```

h5write(obj, file, name, ...)

## Default S3 method:
h5write(
  obj,
  file,
  name,
  createnewfile = TRUE,
  write.attributes = FALSE,
  ...,
  native = FALSE
)

h5writeDataset(obj, h5loc, name, ...)

## S3 method for class 'data.frame'
h5writeDataset(obj, h5loc, name, level = 6, chunk, DataFrameAsCompound = TRUE)

## S3 method for class 'array'
h5writeDataset(
  obj,
  h5loc,
  name,
  index = NULL,
  start = NULL,
  stride = NULL,
  block = NULL,
  count = NULL,
  size = NULL,
  variableLengthString = FALSE,
  encoding = NULL,
  level = 6
)

```

**Arguments**

obj	The R object to be written.
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	The name of the dataset in the HDF5 file.
...	Further arguments passed to <a href="#">H5Dwrite</a> .
createnewfile	If TRUE, a new file will be created if necessary.

<code>write.attributes</code>	(logical) If TRUE, all R-attributes attached to the object <code>obj</code> are written to the HDF5 file.
<code>native</code>	An object of class <code>logical</code> . If TRUE, array-like objects are treated as stored in HDF5 row-major rather than R column-major orientation. Using <code>native = TRUE</code> increases HDF5 file portability between programming languages. A file written with <code>native = TRUE</code> should also be read with <code>native = TRUE</code> .
<code>h5loc</code>	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.
<code>level</code>	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.
<code>chunk</code>	Specifies the number of items to be include in an HDF5 chunk. If left unspecified the defaults is the smaller of: the total number of elements or the number of elements that fit within 4GB of memory. If <code>DataFrameAsCompound=FALSE</code> each row of the <code>data.frame</code> can be consider an "element".
<code>DataFrameAsCompound</code>	If true, 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.
<code>index</code>	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.
<code>start</code>	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> .
<code>stride</code>	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> .
<code>block</code>	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> .
<code>count</code>	The number of blocks to be written. This argument is ignored, if <code>index</code> is not <code>NULL</code> .
<code>size</code>	The length of the fixed-width string data type, when <code>obj</code> is a character vector. If <code>NULL</code> , this is set to the length of the largest string.
<code>variableLengthString</code>	Whether character vectors should be written as variable-length strings into the attributes. If TRUE, <code>size</code> is ignored.
<code>encoding</code>	The encoding of the string data type. Valid options are "ASCII" or "UTF-8".



## Details

Writes an R object to an HDF5 file. If none 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 write subarrays, (so called hyperslabs) to the HDF5 file. The arguments `start`, `stride`, `block`, `count` define the subset of the dataset in the HDF5 file that is to be written to. 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

`h5write` returns 0 if successful.

## Author(s)

Bernd Fischer, Mike Smith

## References

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

## See Also

[h5ls](#), [h5createFile](#), [h5createDataset](#), [rhdf5](#)

## Examples

```
h5File <- tempfile(fileext = ".h5")
h5createFile( h5File )

# write a matrix
B = array(seq(0.1,2.0,by=0.1),dim=c(5,2,2))
attr(B, "scale") <- "liter"
h5write(B, h5File,"B")

# write a submatrix
h5createDataset(h5File, "S", c(5,8), storage.mode = "integer", chunk=c(5,1), level=7)
h5write(matrix(1:5,nr=5,nc=1), file=h5File, name="S", index=list(NULL,1))
```

---

h5\_writeAttribute      *Write an R object as an HDF5 attribute*

---

### Description

Write an R object as an HDF5 attribute

### Usage

```
h5writeAttribute(
  attr,
  h5obj,
  name,
  encoding = NULL,
  cset = NULL,
  variableLengthString = FALSE,
  asScalar = FALSE
)

## S3 method for class 'array'
h5writeAttribute(
  attr,
  h5obj,
  name,
  encoding = NULL,
  cset = NULL,
  variableLengthString = FALSE,
  asScalar = FALSE
)
```

### Arguments

attr	The R object to be written as an HDF5 attribute.
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 to be written.
encoding	The encoding of the string data type. Valid options are "ASCII" and "UTF-8".
cset	<i>Deprecated in favour of the encoding argument.</i>
variableLengthString	Whether character vectors should be written as variable-length strings into the attributes.
asScalar	Whether length-1 attr should be written into a scalar dataspace.

---

rhdf5

*rhdf5: An interface between HDF5 and R*

---

**Description**

The rhdf5 package provides two categories of functions:

- h5 functions are high-level R functions that provide a convenient way of accessing HDF5 files
- H5 functions mirror much of the the HDF5 C API

# Index

- \* **IO**
  - h5\_createAttribute, 77
  - h5\_dump, 85
  - h5\_FileLocking, 88
  - h5\_write, 94
  - h5closeAll, 10
  - h5ls, 37
- \* **file**
  - h5\_createAttribute, 77
  - h5\_dump, 85
  - h5\_FileLocking, 88
  - h5\_write, 94
  - h5closeAll, 10
  - h5ls, 37
- \* **interface**
  - h5\_createAttribute, 77
  - h5\_dump, 85
  - h5\_write, 94
  - h5ls, 37
- \* **programming**
  - h5\_createAttribute, 77
  - h5\_dump, 85
  - h5\_write, 94
  - h5ls, 37
- [, H5IdComponent-method
  - (H5IdComponent-class), 28
- [, H5Ref-method (H5Ref-class), 60
- [<-, H5IdComponent-method
  - (H5IdComponent-class), 28
- \$, H5IdComponent-method
  - (H5IdComponent-class), 28
- \$<-, H5IdComponent-method
  - (H5IdComponent-class), 28
- &, H5IdComponent, character-method
  - (H5IdComponent-class), 28
- as.integer, 29
- c, H5Ref-method (H5Ref-class), 60
- drop, 29
- h5\_createAttribute, 77
- h5\_createDataset, 79
- h5\_createFile, 82
- h5\_createGroup, 83
- h5\_delete, 84
- h5\_deleteAttribute, 85
- h5\_dump, 85
- h5\_errorHandling, 87
- h5\_FileLocking, 88
- h5\_read, 89
- h5\_readAttributes, 92
- h5\_save, 92
- h5\_set\_extent, 93
- h5\_write, 94
- h5\_writeAttribute, 98
- H5Aclose, 5
- H5Acreate, 5
- H5Adelete, 6
- H5Aexists, 6
- H5Aget\_name, 7
- H5Aget\_space, 7
- H5Aget\_type, 8
- H5Aopen, 8
- H5Aopen(), 5, 7–10
- H5Aopen\_by\_idx (H5Aopen), 8
- H5Aopen\_by\_name (H5Aopen), 8
- H5Aread, 9, 92
- H5Awrite, 10
- H5close (H5functions), 24
- H5close(), 10
- h5closeAll, 10
- h5const (h5constants), 11
- h5constants, 11
- h5constType (h5constants), 11
- h5createAttribute (h5\_createAttribute), 77
- h5createDataset, 29, 97
- h5createDataset (h5\_createDataset), 79

- h5createDataset(), [46](#), [78](#), [83](#), [84](#), [96](#)
- h5createFile, [97](#)
- h5createFile(h5\_createFile), [82](#)
- h5createFile(), [78](#), [81](#), [84](#)
- h5createGroup(h5\_createGroup), [83](#)
- h5createGroup(), [78](#), [81](#), [83](#)
- H5Dchunk\_dims, [12](#)
- H5Dclose, [12](#)
- H5Dclose(), [16](#), [60](#)
- H5Dcreate, [13](#), [94](#), [98](#)
- H5Dcreate(), [5](#), [6](#), [8](#), [77](#)
- h5default(h5constants), [11](#)
- h5delete(h5\_delete), [84](#)
- h5deleteAttribute(h5\_deleteAttribute), [85](#)
- H5Dget\_create\_plist, [14](#)
- H5Dget\_space, [14](#)
- H5Dget\_space(), [5](#), [13](#), [17](#)
- H5Dget\_storage\_size, [15](#)
- H5Dget\_type, [15](#)
- h5disableFileLocking(h5\_FileLocking), [88](#)
- H5Dopen, [16](#), [94](#), [98](#)
- H5Dopen(), [5](#), [6](#), [8](#), [77](#)
- H5Dread, [17](#), [90](#)
- H5Dset\_extent, [18](#)
- h5dump(h5\_dump), [85](#)
- h5dump(), [38](#), [93](#)
- H5Dwrite, [19](#), [95](#)
- h5enableFileLocking(h5\_FileLocking), [88](#)
- h5errorHandling(h5\_errorHandling), [87](#)
- H5Fclose, [19](#)
- H5Fcreate, [20](#), [88](#), [90](#), [94–96](#), [98](#)
- H5Fcreate(), [5](#), [6](#), [8](#), [13](#), [19](#), [21](#), [22](#), [37](#), [77](#), [79](#), [83](#), [86](#), [92](#)
- H5Fflush, [20](#)
- H5Fget\_access\_plist(H5Fget\_plist), [22](#)
- H5Fget\_create\_plist(H5Fget\_plist), [22](#)
- H5Fget\_filesize, [21](#)
- H5Fget\_name, [21](#)
- H5Fget\_plist, [22](#)
- H5Fis\_hdf5, [23](#)
- H5Fopen, [23](#), [29](#), [90](#), [94–96](#), [98](#)
- H5Fopen(), [5](#), [6](#), [8](#), [13](#), [19](#), [21](#), [22](#), [37](#), [77](#), [79](#), [83](#), [86](#), [92](#)
- H5functions, [24](#)
- H5garbage\_collect(H5functions), [24](#)
- H5Gclose, [25](#)
- H5Gclose(), [28](#)
- H5Gcreate, [25](#), [90](#), [94–96](#), [98](#)
- H5Gcreate(), [5](#), [6](#), [8](#), [13](#), [25](#), [26](#), [37](#), [77](#), [79](#), [83](#), [86](#), [92](#)
- H5Gcreate\_anon, [26](#), [41](#)
- H5get\_libversion(H5functions), [24](#)
- H5Gget\_info, [26](#)
- H5Gget\_info\_by\_idx(H5Gget\_info), [26](#)
- H5Gget\_info\_by\_name(H5Gget\_info), [26](#)
- H5Gopen, [27](#), [90](#), [94–96](#), [98](#)
- H5Gopen(), [5](#), [6](#), [8](#), [13](#), [25](#), [37](#), [77](#), [79](#), [83](#), [86](#), [92](#)
- H5IdComponent, [5–10](#), [12–23](#), [25–28](#), [30–37](#), [39–58](#), [63–72](#), [77](#), [79](#), [83](#), [86](#), [90](#), [92](#), [94–96](#), [98](#)
- H5IdComponent-class, [28](#)
- H5Iget\_name, [29](#)
- H5Iget\_type, [30](#)
- H5Iis\_valid, [31](#)
- H5Lcopy, [31](#)
- H5Lcreate\_external, [32](#)
- H5Ldelete, [33](#)
- H5Lexists, [34](#)
- H5Lget\_info, [34](#)
- h5listIdentifier(h5listObjects), [35](#)
- h5listObjects, [35](#)
- H5Lmove, [36](#)
- h5ls, [37](#), [91](#), [97](#)
- h5ls(), [86](#), [93](#)
- H5Oclose, [38](#)
- H5Oclose(), [42](#), [60](#)
- H5Ocopy, [39](#)
- H5Oget\_num\_attrs, [40](#)
- H5Oget\_num\_attrs\_by\_name(H5Oget\_num\_attrs), [40](#)
- H5Olink, [41](#)
- H5Olink(), [26](#)
- H5Oopen, [42](#)
- H5Oopen(), [39](#)
- H5open(H5functions), [24](#)
- H5P\_chunk, [54](#)
- H5P\_chunk\_cache, [55](#)
- H5P\_create\_intermediate\_group, [55](#)
- H5P\_fill\_time, [56](#), [57](#)
- H5P\_fill\_value, [56](#)
- H5P\_layout, [57](#)
- H5P\_libver\_bounds, [58](#)
- H5Pall\_filters\_avail, [43](#)

- H5Pclose, [43](#)
- H5Pcopy, [44](#)
- H5Pcopy(), [20](#)
- H5Pcreate, [44](#)
- H5Pcreate(), [20](#)
- H5Pfill\_value\_defined, [45, 57](#)
- H5Pget\_chunk (H5P\_chunk), [54](#)
- H5Pget\_class, [45](#)
- H5Pget\_create\_intermediate\_group  
(H5P\_create\_intermediate\_group),  
[55](#)
- H5Pget\_fill\_time (H5P\_fill\_time), [56](#)
- H5Pget\_filter (H5Pall\_filters\_avail), [43](#)
- H5Pget\_istore\_k (H5Pset\_istore\_k), [49](#)
- H5Pget\_layout (H5P\_layout), [57](#)
- H5Pget\_libver\_bounds  
(H5P\_libver\_bounds), [58](#)
- H5Pget\_nfilters (H5Pall\_filters\_avail),  
[43](#)
- H5Pget\_obj\_track\_times  
(H5Pobject\_track\_times), [46](#)
- H5Pget\_shared\_mesg\_index  
(H5Pset\_shared\_mesg\_index), [50](#)
- H5Pget\_shared\_mesg\_nindexes  
(H5Pset\_shared\_mesg\_nindexes),  
[51](#)
- H5Pget\_shared\_mesg\_phase\_change  
(H5Pset\_shared\_mesg\_phase\_change),  
[51](#)
- H5Pget\_sizes (H5Pset\_sizes), [52](#)
- H5Pget\_sym\_k (H5Pset\_sym\_k), [53](#)
- H5Pget\_userblock (H5Pset\_userblock), [54](#)
- H5Pget\_version, [46](#)
- H5Pobject\_track\_times, [46](#)
- H5Pset\_blosc, [47](#)
- H5Pset\_bzip2, [47](#)
- H5Pset\_chunk (H5P\_chunk), [54](#)
- H5Pset\_chunk\_cache (H5P\_chunk\_cache), [55](#)
- H5Pset\_create\_intermediate\_group  
(H5P\_create\_intermediate\_group),  
[55](#)
- H5Pset\_deflate, [48](#)
- H5Pset\_fapl\_ros3, [48](#)
- H5Pset\_fill\_time (H5P\_fill\_time), [56](#)
- H5Pset\_fill\_value (H5P\_fill\_value), [56](#)
- H5Pset\_istore\_k, [49](#)
- H5Pset\_layout (H5P\_layout), [57](#)
- H5Pset\_layout(), [54](#)
- H5Pset\_libver\_bounds  
(H5P\_libver\_bounds), [58](#)
- H5Pset\_lzf, [49](#)
- H5Pset\_nbit, [50](#)
- H5Pset\_obj\_track\_times  
(H5Pobject\_track\_times), [46](#)
- H5Pset\_shared\_mesg\_index, [50](#)
- H5Pset\_shared\_mesg\_nindexes, [51](#)
- H5Pset\_shared\_mesg\_phase\_change, [51](#)
- H5Pset\_shuffle, [52](#)
- H5Pset\_sizes, [52](#)
- H5Pset\_sym\_k, [53](#)
- H5Pset\_szip, [53](#)
- H5Pset\_userblock, [54](#)
- H5R, [58](#)
- H5Rcreate, [59](#)
- H5Rdereference, [60](#)
- h5read (h5\_read), [89](#)
- h5read(), [78, 81, 83, 84, 86, 93](#)
- h5readAttributes (h5\_readAttributes), [92](#)
- H5Ref, [59](#)
- H5Ref-class, [60](#)
- H5Rget\_name, [61](#)
- H5Rget\_obj\_type, [62](#)
- H5Rget\_region, [62](#)
- h5save (h5\_save), [92](#)
- H5Sclose, [63](#)
- H5Sclose(), [62](#)
- H5Scombine\_hyperslab, [63](#)
- H5Scombine\_hyperslab(), [65, 69, 70](#)
- H5Scombine\_select, [64](#)
- H5Scombine\_select(), [64](#)
- H5Scopy, [65](#)
- H5Screate, [66, 67](#)
- H5Screate(), [5, 13, 17, 63](#)
- H5Screate\_simple, [66, 67, 72](#)
- H5Screate\_simple(), [5, 13, 17](#)
- h5set\_extent (h5\_set\_extent), [93](#)
- H5Sget\_select\_npoints, [67](#)
- H5Sget\_simple\_extent\_dims, [68](#)
- H5Sis\_simple, [68](#)
- H5Sselect\_all, [68](#)
- H5Sselect\_hyperslab, [69](#)
- H5Sselect\_hyperslab(), [64](#)
- H5Sselect\_index, [70](#)
- H5Sselect\_none, [71](#)
- H5Sselect\_valid, [71](#)
- H5Sset\_extent\_simple, [72](#)

H5Sunlimited, [72](#)  
H5T\_cset, [74](#)  
H5T\_precision, [74](#)  
H5T\_size, [75](#)  
H5T\_strpad, [75](#)  
H5Tcopy, [73](#)  
h5testFileLocking (h5\_FileLocking), [88](#)  
H5Tget\_cset (H5T\_cset), [74](#)  
H5Tget\_precision (H5T\_precision), [74](#)  
H5Tget\_size (H5T\_size), [75](#)  
H5Tget\_strpad (H5T\_strpad), [75](#)  
H5Tis\_variable\_str, [73](#)  
H5Tset\_cset (H5T\_cset), [74](#)  
H5Tset\_precision (H5T\_precision), [74](#)  
H5Tset\_size (H5T\_size), [75](#)  
H5Tset\_strpad (H5T\_strpad), [75](#)  
h5validObjects (h5listObjects), [35](#)  
h5version, [76](#)  
h5write (h5\_write), [94](#)  
h5write(), [78](#), [81](#), [83](#), [84](#), [93](#)  
h5writeAttribute (h5\_writeAttribute), [98](#)  
h5writeAttribute(), [78](#)  
h5writeDataset (h5\_write), [94](#)  
H5Zfilter\_avail, [76](#)  
  
length, H5Ref-method (H5Ref-class), [60](#)  
  
rhdf5, [78](#), [83](#), [87](#), [97](#), [99](#)  
  
show, H5IdComponent-method  
    (H5IdComponent-class), [28](#)  
show, H5Ref-method (H5Ref-class), [60](#)