HDF5 C++ User's Notes

This User's Note provides an overview of the structure, the availability, and the limitations of the C++ API of HDF5. It lists the classes and member functions included in the API and provides some examples of their applications. The C++ API is itself under development and does not have a complete User's Guide or Reference Manual. In addition, it is assumed that the reader has knowledge of the HDF5 file format and its components. For a complete User's Guide and Reference Manual of HDF5, please refer to the HDF home page at http://hdf.ncsa.uiuc.edu. At this time, to effectively utilize the C++ API, please refer to the C++ Interface, off of the Reference Manual of HDF5 page.

The User's Note includes an Overview section that gives the overall structure of the API. Following the Overview section is the Class Description section that briefly describes the classes and the functions they provide. This section also lists the limitations of the current version and describes plans for improvement/completion of some of the classes/functions. The final section, Examples, describes the examples that are provided with the source code distribution.

1. Overview

The HDF5 C++ API consists of the classes listed in the table below. All classes are included in a namespace called H5.

Class	Description
`H5Library`	`provides general-purpose library functions`
`IdComponent`	`is a base class that manage HDF5 object identifier`
`RefCounter`	`provides reference counting mechanism`
`CommonFG`	`is a base class for commonalities of H5File and Group`
`H5File`	`provides functions that access an HDF5 file`
`H5Object`	`base class for commonalties of all HDF5 objects which include groups, datasets, datatypes, and attributes`
`Group`	`is an H5Object; provides functions that access HDF5 groups`
`AbstractDs`	`base class for commonalities of DataSet and Attribute`
`DataSet`	`provides functions that access a dataset`
`Attribute`	`provides functions that access an attribute`
`DataType`	`is an H5Object; provides functions that access a general datatype, which can be an enumeration datatype, compound datatype, or atomic datatype`
`EnumType`	`is a DataType; provides functions that access an enumeration datatype`
`CompType`	`is a DataType; provides functions that access a compound datatype`
`AtomType`	`is a DataType; base class for commonalties of HDF5 predefined provides functions that access an enumeration datatype`
`PredType`	`is an AtomType; provides the constant PredType objects for all the predefined datatype provided by the HDF5 library`
`IntType`	`is an AtomType; provides functions that access an integer datatype`
`FloatType`	`is an AtomType; provides functions that access an`

	floating-point datatype
StrType	is an AtomType; provides functions that access an string datatype
DataSpace	provides functions that access the HDF5 dataspace
PropList	provides common accesses to the property lists
DSetCreatPropList	is a PropList; provides accesses to a dataset creation property list
DSetMemXferPropList	is a PropList; provides accesses to a dataset memory and transfer property list
FileAccPropList	is a PropList; provides accesses to a file access property list
FileCreatPropList	is a PropList; provides accesses to a file creation property list
Exception	provides the mechanism for handling errors returned by the C HDF5 library; it has several subclasses for specific exceptions

Figure 1 shows the hierarchical relationship between the classes.

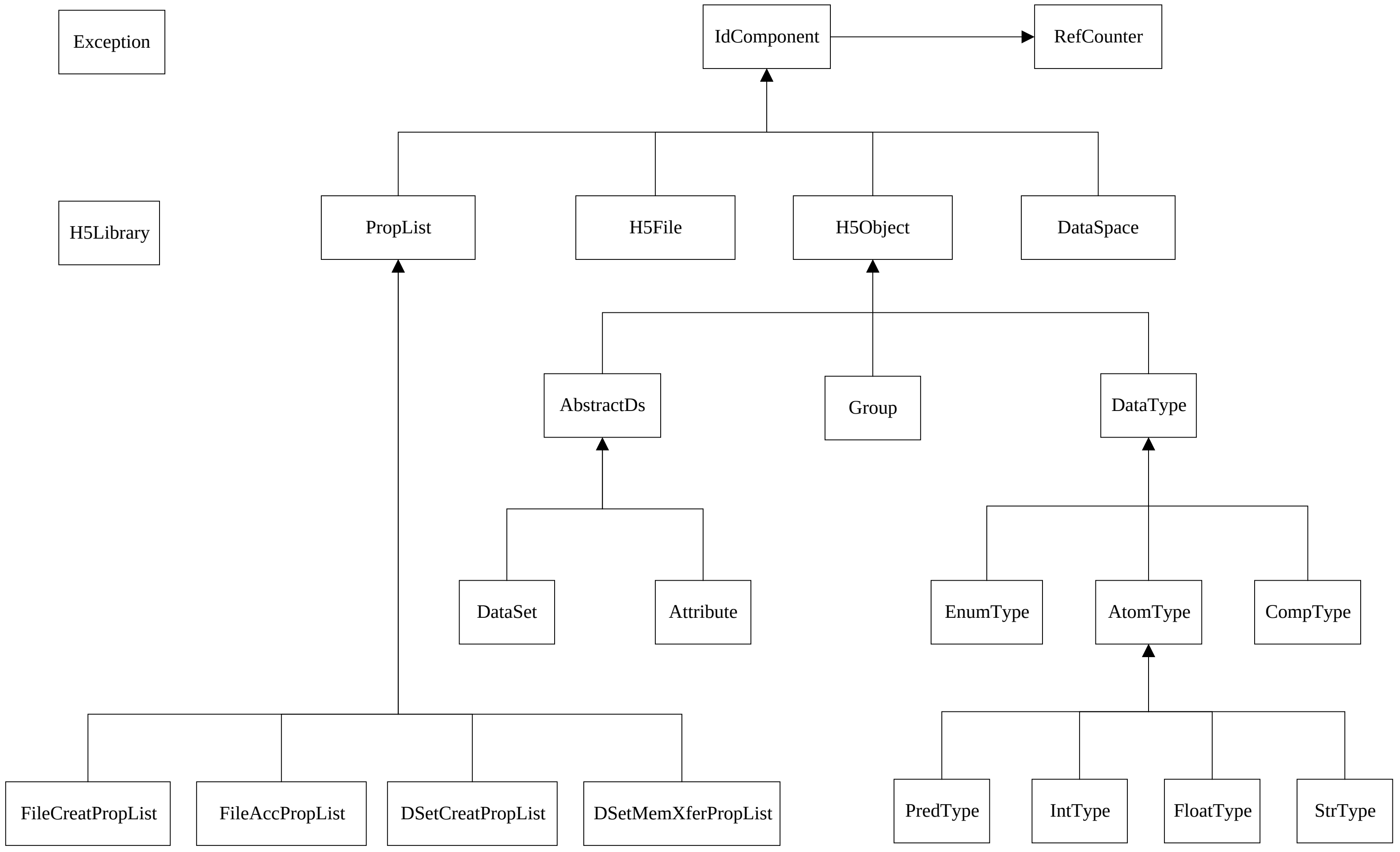

Figure 1. Class hierarchy of HDF5 C++ API

The figure shows that all the classes in the API, except H5Library and Exception, inherit from the class IdComponent. The elements that are represented by these classes are identified by an identifier that is defined and manipulated by the HDF5 library. IdComponent relieves the C++ API users from the concern about properly managing the identifiers of any HDF5 objects. The figure also illustrates the inheritance relationship between the subclasses of IdComponent.

H5Library is a stand-alone class to provide accesses to the library as a whole. Although Exception is also showed as having no inheritance relationship, in fact, it has many subclasses. These subclasses are for the specific exceptions and have no additional members. Thus, they are listed in the section about Exception class but not shown in the diagram. Another aspect that is not shown in the figure is that the classes H5File and Group also inherit from another base class, CommonFG, because of their commonality that does not exist in other subclasses of IdComponent.

The classes of the API and their members are described in the subsequent sections.

2. Classes Description

This section briefly describes the classes that form the C++ API of HDF5. Each subsection below gives a brief description of a class and provides a table that lists the public services that the class provides. Where necessary, the subsection also indicates the limitations of the current implementation of the described class and/or plans for its improvement or completion.

2.1. H5Library

This class provides some services that are used to access the HDF5 library. It is independent with other classes in the API. Its member functions are static so there is no need for an instance of this class to exist to use them.

Member Function	Purpose
open	Initializes the HDF5 library
close	Flushes all data to disk and clean up resources
dontAtExit	Instructs HDF5 C library not to install atexit clean up routine; this is helpful when having global objects in the application because the clean up routine might be executed before the global destructors and prematurely close any needed HDF5 components
getLibVersion	Retrieves the HDF library release numbers
checkVersion	Verifies that the arguments match the version numbers compiled into the library

2.2. Exception

This class provides services to support user exception handling. All HDF5 C++ API calls that throw exceptions provide an instance of a class derived from Exception as a parameter. Thus the user can extract runtime information from it through the use of a corresponding catch procedure. Currently, Exception is used to derive the subclasses that support specific types of HDF5 errors that are generated by the C APIs, including H5F, H5G, H5S, H5T, H5P, H5D, and H5A. All of the functionality provided by these subclasses is inherited from Exception. These subclasses are listed below:

- FileIException for errors generated by the C API H5F
- GroupIException for errors generated by the C API H5G

- DataSpaceIException for errors generated by the C API H5S
- DataTypeIException for errors generated by the C API H5T
- PropListIException for errors generated by the C API H5P
- DataSetIException for errors generated by the C API H5D
- AttributeIException for errors generated by the C API H5A
- LibraryIException for errors generated by the C API H5

The following table lists the services provided by Exception.

Member Function	Purpose
Exception	Default constructor
Exception	Constructor that stores a given detailed message
Exception	Copy constructor
getMajorString	Returns the character string that describes an error specified by a major error number
getMinorString	Returns the minor error string of the exception
getFuncName	Returns the character string that describes an error specified by a minor error number
getFileName	Returns the name of the file in which the error occurs
getDescString	Returns the description string provided by the HDF5 library described the nature of the error
getLine	Returns the line number where the error occurs
getDetailMesg	Returns the user’s detailed message annotating the error
setAutoPrint	Turns on the automatic error printing
dontPrint	Turns off the automatic error printing
getAutoPrint	Retrieves the current settings for the automatic error stack traversal function and its data
clearErrorStack	Clears the error stack for the current thread
walkErrorStack	Walks the error stack for the current thread, calling the specified function
walkDefErrorStack	Default error stack traversal callback function that prints error messages to the specified output stream
printError	Displays the error information in the default manner - Note: *this function will be made virtual in the next release*
~Exception	*missing destructor; will be virtual*

2.3. IdComponent

This class provides a mean to ensure proper use of and to manage reference counting for an identifier of any HDF5 object. Hence, all HDF5 component classes benefit from this class. IdComponent uses RefCounter for its reference counting mechanism.

Member Function	Purpose
setId	Sets the identifier of this instance to a new value
getId	Gets the identifier of the instance, i.e. the current HDF5 object identifier
incRefCount	Increment reference counter
decRefCount	Decrement reference counter
getCounter	Gets the reference counter to this identifier
noReference	Determines whether there is no more reference to this identifier; note: the reference counter is decremented before checking

reset	Resets this instance by deleting its reference counter of the old identifier; this instance can then be used for another HDF5 identifier
IdComponent	Constructor that takes an HDF5 identifier
IdComponent	Copy constructor
~IdComponent	Virtual destructor - *has a bug*

2.4. RefCounter

RefCounter provides a reference counting mechanism. IdComponent uses this class to keep track of the number of copies of an HDF5 object so that the object's identifier can be properly released.

Member Function	Purpose
getCounter	Returns the value of the counter
noReference	Determines whether the counter is back to 0; note: the counter is decremented before checking
increment	Increment the counter
decrement	Decrement the counter
RefCounter	Default constructor
~RefCounter	Destructor - *will be virtual*

2.5. CommonFG

CommonFG means commonality between file and group. This class is a protocol class. Its existence is simply to provide the common services that are provided by H5File and Group. The file or group in the context of this class is referred to as 'location'.

Member Function	Purpose
createGroup	Creates a new group at this location
openGroup	Opens an existing group at this location
createDataSet	Creates a new dataset at this location
openDataSet	Opens a existing dataset at this location
openDataType	Opens a named generic datatype at this location
openEnumType	Opens a named enumeration datatype at this location
openCompType	Opens a named compound datatype at this location
openIntType	Opens a named integer datatype at this location
openFloatType	Opens a named floating-point datatype at this location
openStrType	Opens a named string datatype at this location
link	Creates a link of the specified type from a new name to the current name; both names are interpreted relative to this location.
unlink	Removes the specified name at this location
move	Renames an object within this location
getObjinfo	Retrieves information about an object, given its name and link, at this location
getLinkval	Returns the name of the HDF5 object that the given symbolic link points to
setComment	Sets the comment for an object, specified by its name, in this location
getComment	Gets the comment of an object, specified by its name, in this location
mount	Mounts a file, specified by its name, to this location
unmount	Unmounts a file, specified by its name, from this location

throwException	pure virtual - implemented by H5File and Group so that each class can throw the appropriate exception when an error occurs within CommonFG
CommonFG	Default constructor
~CommonFG	Virtual default constructor

2.6. H5File

This class uses a number of functions, which are publicly provided by class CommonFG, to access HDF5 files. In the context of these functions, a file is considered a location. Refer to Section 2.5 for the mentioned functions. In addition, H5File provides some functions that are specific to HDF5 files and not applicable to group. These functions are listed in the following table.

Member Function	Purpose
H5File	Creates or opens an HDF5 file
H5File	Copy constructor
isHdf5	Determines if a file, specified by its name, is in HDF5 format
reopen	Reopens this file
getCreatePlist	Gets the creation property list of this file
getAccessPlist	Gets the access property list of this file
throwException	throws FileIException
~H5File	Virtual destructor

2.7. H5Object

This class provides services that are used to access an HDF5 object, which can be a group, a dataset, an attribute, or a named datatype.

Member Function	Purpose
H5Object	Copy constructor
flush	Flushes all buffers associated with this object, which belongs to a file, to disk
createAttribute	Creates an attribute for this object, which can be a group, a dataset, or a named datatype
openAttribute	Opens an attribute for this object given the attribute’s name or index; the object can be either a group, a dataset, or a named datatype
iterateAttrs	Iterate user’s function over the attributes of this object
getNumAttrs	Determines the number of attributes attached to this object
removeAttr	Removes the named attribute from this object
~H5Object	Virtual destructor

2.8. Group

Group represents the HDF5 group. As with H5File, this class inherits from CommonFG those functions that access an HDF5 group, which is called location in that context. It also inherits from another base class, H5Object, those characteristics of an HDF5 object, namely, the functions that access HDF5 attributes.

Member Function	Purpose
`Group`	`Default constructor`
`Group`	`Copy constructor`
`Group`	`Constructor that takes an HDF5 identifier`
`iterateElems`	`Iterates over the elements of this group -` *`C++ style version not yet implemented`*
`throwException`	`throw GroupIException`
	`Default constructor and copy constructor`
`~Group`	`Virtual destructor`

2.9. AbstractDs

AbstractDs is from the term abstract dataset. Because an HDF5 attribute is similar to a dataset, this abstract dataset class is introduced to provide their common functionality. AbstractDs is an abstract base class, from which the classes Attribute and DataSet are derived. This class also publicily inherits from H5Object and passes down the services that H5Object provides.

Member Function	Purpose
`AbstractDs`	`Copy constructor`
`getSpace`	`Gets the dataspace of this dataset - pure virtual`
`getTypeClass`	`Gets the class of the datatype that is used by this dataset`
`getDataType`	`Gets the generic datatype of a dataset or an attribute.`
`getEnumType`	`Gets the enumeration datatype of a dataset or an attribute.`
`getCompType`	`Gets the compound datatype of a dataset or an attribute.`
`getIntType`	`Gets the integer datatype of a dataset or an attribute.`
`getFloatType`	`Gets the floating-point datatype of a dataset or an attribute.`
`getStrType`	`Gets the string datatype of a dataset or an attribute.`
`~AbstractDs`	`Virtual destructor`

Notes on implementation:

2.9.1. `getSpace` is a pure virtual function. DataSet and Attribute provide their own implementation.

2.9.2. In the next version of the C++ API, the DataSet and Attribute member functions read and write might be overloaded and moved up into this base class.

2.10. DataSet

This class provides the services that are used to access an HDF5 dataset.

Member Function	Purpose
`DataSet`	`Default constructor`
`DataSet`	`Copy constructor`
`getCreatePlist`	`Gets the creation property list of this dataset`
`getStorageSize`	`Gets the storage size of this dataset`
`read`	`Reads the data of this dataset and stores it in the provided buffer.  The memory and file dataspaces and the transferring property list can be defaults.`
`write`	`Writes the buffered data to this dataset.  The memory and file dataspaces and the transferring property list can be defaults`
`iterateElems`	`Iterates over all selected elements in a dataspace -` *`C++ style version not yet implemented`*
`extend`	`Extends the dataset with unlimited dimension`
`~DataSet`	`Virtual destructor`

Notes on implementation:

2.10.1. `read` and `write` may be implemented using operators >> and << in the next version of the C++ API.

2.10.2. `iterateElems` is not yet implemented. It may be moved to class DataSpace since it is iterating over elements that are in a dataspace.

2.11. Attribute

This class provides the services that are used to access an HDF5 object's attribute.

Member Function	Purpose
`Attribute`	`Constructor that takes an HDF5 identifier`
`Attribute`	`Copy constructor`
`read`	`Reads data from this attribute`
`write`	`Writes data to this attribute`
`getName`	`Gets the name of this attribute`
`~Attribute`	`Virtual destructor`

Notes on implementation:

2.11.1. `read` and `write` may be implemented using operators >> and << in the next version of the C++ API.

2.12. DataType

This class provides the services that are used to access an HDF5 generic datatype. Several subclasses are derived from DataType.

Member Function	Purpose
`DataType`	`Default constructor`
`DataType`	`Copy constructor`
`DataType`	`Constructor that takes an existing identifier`
`DataType`	`Constructor that takes a datatype’s class and size`
`copy`	`Copies an existing datatype to this datatype instance`

getClass	Returns the datatype class identifier
commit	Commits a transient datatype to a file, creating a new named datatype
committed	Determines whether a datatype is a named type or a transient type
find	Finds a conversion function that can handle the conversion this datatype to the given datatype, dest.
convert	Converts data from between specified datatypes
setOverflow	Sets the overflow handler to a specified function
getOverflow	Returns a pointer to the current global overflow function
lock	Locks a datatype
getSize	Returns the size of a datatype
getSuper	Returns the base datatype from which a datatype is derived *- not completely implemented yet*
registerFunc	Registers a conversion function
unregister	Removes a conversion function from all conversion paths
setTag	Tags an opaque datatype
getTag	Gets the tag associated with an opaque datatype
~DataType	Virtual destructor

Notes on implementation:

2.12.1. Following is the structure of the subclasses of DataType. Those that are in italic face are not yet implemented.

DataType
- CompType: is a compound datatype.
- EnumType: is an enumeration datatype.
- AtomType: is an atomic datatype and has the following subclasses.
  - PredType: is a predefined datatype for integer, float, and string. Note that this class may need inherit directly from DataType; more study is necessary.
  - *Reference*: is predefined datetype for object and region references and is not yet implemented. Note that once this class is implemented an intermediate base class might be introduced for PredType and Reference.
  - IntType: is a user-defined integer datatype.
  - FloatType: is a user-defined floating-point datatype.
  - StrType: is a user-defined string datatype.
  - *BitFieldType*: is a user-defined bitfield datatype and is not yet implemented.
  - *OpaqueType*: is a user-defined opaque datatype and is not yet implemented.

2.12.2. The function getSuper currently only returns the generic datatype. To get the specific datatype, the user must cast it. In future versions, its implementation will be improved.

2.13. EnumType

This class provides the services that are used to access an enumeration datatype. It is derived from DataType.

Member Function	Purpose
EnumType	Default constructor
EnumType	Copy constructor
EnumType	Creates a new enumeration datatype based on a native signed integer type, whose size is given by size
EnumType	Gets the enumeration datatype of the specified dataset
EnumType	Creates a new enum datatype based on an integer datatype
insert	Inserts a new member to this enumeration type
nameOf	Returns the symbol name corresponding to a specified member of this enumeration datatype
valueOf	Returns the value corresponding to a specified member of this enumeration datatype
getMemberValue	Returns the value of an enumeration datatype member
~EnumType	Virtual destructor

2.14. CompType

This class provides the services that are used to access a compound datatype. It is derived from DataType.

Member Function	Purpose
CompType	Default constructor
CompType	Copy constructor
CompType	Creates a new compound datatype given its size
CompType	Gets the compound datatype of the specified dataset
getNmembers	Gets the number of members in this compound datatype
getMemberName	Gets the name of a member of this compound datatype
getMemberOffset	Gets the offset of a member of this compound datatype
getMemberDims	Gets the dimensionality of the specified member
getMemberClass	Gets the type class identifier of the specified member
getMemberDataType	Gets the generic datatype of the specified member in this compound datatype; the subsequent functions are for the specific sup-types.
getMemberEnumType	Gets the enumeration datatype of a dataset or an attribute.
getMemberCompType	Gets the compound datatype of a dataset or an attribute.
getMemberIntType	Gets the integer datatype of a dataset or an attribute.
getMemberFloatType	Gets the floating-point datatype of a dataset or an attribute.
getMemberStrType	Gets the string datatype of a dataset or an attribute.
insertMember	Adds a new member to this compound datatype; *the ability to insert an array is removed from this member function*
pack	Recursively removes padding from within this compound datatype

2.15. AtomType

This class is derived from DataType and, in addition, provides common services to access predefined types, integer type, floating-point type, and string type.

Member Function	Purpose
`AtomType`	Copy constructor
`setSize`	Sets the total size for an atomic datatype
`getOrder`	Returns the byte order of an atomic datatype
`setOrder`	Sets the byte ordering of an atomic datatype
`getPrecision`	Returns the precision of an atomic datatype
`setPrecision`	Sets the precision of an atomic datatype
`getOffset`	Retrieves the bit offset of the first significant bit
`setOffset`	Sets the bit offset of the first significant bit
`getPad`	*temporarily removed from this class*
`setPad`	*temporarily removed from this class*
`~AtomType`	Virtual destructor

2.16. PredType

This class contains the definition of objects that correspond to the predefined datatypes defined in the HDF5 library. Refer to the header file PredType.h for specific names.

2.17. IntType

This class provides the services used to access the user-defined integer datatype. It is derived from AtomType.

Member Function	Purpose
`IntType`	Default constructor
`IntType`	Copy constructor
`IntType`	Creates an IntType using a predefined integer type
`IntType`	Gets the integer datatype of the specified dataset
`getSign`	Returns the sign type for an integer type
`setSign`	Sets the sign property for an integer type
`~IntType`	Virtual destructor

2.18. FloatType

This class provides the services used to access the user-defined floating-point datatype. It is derived from AtomType.

Member Function	Purpose
`FloatType`	Default constructor
`FloatType`	Copy constructor
`FloatType`	Creates a new FloatType using a predefined floating-point type
`FloatType`	Gets the floating-point datatype of the specified dataset
`getFields`	Retrieves floating point datatype bit field information
`setFields`	Sets locations and sizes of floating point bit fields
`getEbias`	Retrieves the exponent bias of a floating-point type
`setEbias`	Sets the exponent bias of a floating-point type
`getNorm`	Returns the mantissa normalization of a floating-point datatype
`setNorm`	Sets the mantissa normalization of a floating-point

	datatype
getInpad	Retrieves the internal padding type for unused bits in floating-point datatypes
setInpad	Fills unused internal floating point bits
~FloatType	Virtual destructor

2.19. StrType

This class provides the services used to access the user-defined string datatype. It is derived from AtomType.

Member Function	Purpose
StrType	Default constructor
StrType	Copy constructor
StrType	Creates a new StrType datatype using a predefined string type
StrType	Gets the string datatype of the specified dataset
getCset	Returns the character set type of this string datatype
setCset	Sets character set to be used
getStrpad	Retrieves the string padding method for this string datatype
setStrpad	Defines the storage mechanism for character strings
~StrType	Virtual destructor

2.20. DataSpace

This class provides services that are used to access an HDF5 dataspace. It inherits the HDF5 object identifier management from the base class IdComponent.

Member Function	Purpose
DataSpace	Default constructor
DataSpace	Copy constructor
DataSpace	Creates a dataspace object given the space type
DataSpace	Creates a simple dataspace
copy	Makes copy of an existing dataspace instance
isSimple	Determines if this dataspace is a simple one
offsetSimple	Sets the offset of this simple dataspace
getSimpleExtentDims	Retrieves dataspace dimension size and maximum size
getSimpleExtentNdims	Gets the dimensionality of this dataspace
getSimpleExtentNpoints	Gets the number of elements in this dataspace
getSimpleExtentType	Gets the current class of this dataspace
extentCopy	Copies the extent of this dataspace
setExtentSimple	Sets or resets the size of this dataspace
setExtentNone	Removes the extent from this dataspace
getSelectNpoints	Gets the number of elements in this dataspace selection
getSelectHyperNblocks	Get number of hyperslab blocks
getSelectHyperBlocklist	Gets the list of hyperslab blocks currently selected
getSelectElemNpoints	Gets the number of element points in the current selection
getSelectElemPointlist	Retrieves the list of element points currently selected
getSelectBounds	Gets the bounding box containing the current

	selection
selectElements	Selects array elements to be included in the selection
selectAll	Selects the entire dataspace
selectNone	Resets the selection region to include no elements
selectValid	Verifies that the selection is within the extent of the dataspace
selectHyperslab	Selects a hyperslab region to add to the current selected region

Notes on implementation:

2.20.1. In the next version of the C++ API, this class may be broken into a class hierarchy that reflects the nature of the dataspace.

2.21. PropList

This class provides the services used to access the HDF5 file and data set property lists. It inherits the HDF5 object identifier management from the base class IdComponent.

Member Function	Purpose
PropList	Default constructor
PropList	Copy constructor
PropList	Creates a property list given its type
copy	Makes a copy of the given property list
getClass	Gets the type of the property list, i.e, H5P_FILE_CREATE, H5P_FILE_ACCESS, etc…
~PropList	Virtual destructor

2.22. FileCreatPropList

This class is derived from class PropList. It also provides the services specifically used to access the HDF5 file creation property lists.

Member Function	Purpose
FileCreatPropList	Default constructor
FileCreatPropList	Copy constructor
FileCreatPropList	Creates a file creation property list
getVersion	Retrieves version information for various parts of a file.
setUserblock	Sets the userblock size field of a file creation property list.
getUserblock	Gets the size of a user block in this file creation property list.
setSizes	Sets file size-of addresses and sizes.
getSizes	Retrieves the size-of address and size quantities stored in a file according to this file creation property list.
setSymk	Sets the size of parameters used to control the symbol table nodes.
setIstorek	Sets the size of parameter used to control the B-trees for indexing chunked datasets.
getIstorek	Returns the 1/2 rank of an indexed storage B-tree.
~FileCreatPropList	Virtual destructor

2.23. FileAccPropList

This class is derived from class PropList. It also provides the services specifically used to access the HDF5 file access property lists.

Member Function	Purpose
`FileAccPropList`	`Default constructor`
`FileAccPropList`	`Copy constructor`
`FileAccPropList`	`Creates a file access property list`
`setCache`	`Sets the meta data cache and raw data chunk cache parameters.`
`getCache`	`Retrieves maximum sizes of data caches and the preemption policy value.`
`setAlignment`	`Sets alignment properties of this file access property list.`
`getAlignment`	`Retrieves the current settings for alignment properties from this file access property list.`
`setGcReferences`	`Sets garbage collecting references flag`
`getGcReferences`	`Returns garbage collecting references setting.`
`~FileAccPropList`	`Virtual destructor`
`setStdio`	*The following member functions were removed since parallel mode is not supported by C++ API*
`getStdio`	*removed*
`getDriver`	*removed*
`setSec2`	*removed*
`getSec2`	*removed*
`setCore`	*removed*
`getCore`	*removed*
`setFamily`	*removed*
`getFamily`	*removed*
`setSplit`	*removed*
`getSplit`	*removed*

2.24. DSetCreatPropList

This class is derived from class PropList. It also provides the services specifically used to access the HDF5 dataset creation property lists.

Member Function	Purpose
`DSetCreatPropList`	`Default constructor`
`DSetCreatPropList`	`Copy constructor`
`DSetCreatPropList`	`Creates a dataset creation property list`
`setLayout`	`Sets the type of storage used to store the raw data for the dataset that uses this property list`
`getLayout`	`Gets the layout of the raw data storage of the data that uses this property list`
`setChunk`	`Sets the size of the chunks used to store a chunked layout dataset.`
`getChunk`	`Retrieves the size of the chunks used to store a chunked layout dataset.`
`setDeflate`	`Sets compression method and compression level`
`setFillValue`	`Sets a dataset fill value`
`getFillValue`	`Retrieves a dataset fill value`
`setFilter`	`Adds a filter to the filter pipeline`
`getNfilters`	`Returns the number of filters in the pipeline`

getFilter	Returns information about a filter in a pipeline
setExternal	Adds an external file to the list of external files
getExternalCount	Returns the number of external files for a dataset
getExternal	Returns information about an external file
~DSetCreatPropList	Virtual destructor

2.25. DSetMemXferPropList

This class is derived from class PropList. It also provides the services specifically used to access the HDF5 data set memory and transfer property lists.

Member Function	Purpose
DSetMemXferPropList	Default constructor
DSetMemXferPropList	Copy constructor
DSetMemXferPropList	Creates a dataset memory and transfer property list
setBuffer	Sets type conversion and background buffers
getBuffer	Reads buffer settings
setPreserve	Sets the dataset transfer property list status to TRUE or FALSE
getPreserve	Checks status of the dataset transfer property list
setHyperCache	Indicates whether to cache hyperslab blocks during I/O
getHyperCache	Returns information regarding the caching of hyperslab blocks during I/O
setBtreeRatios	Sets B-tree split ratios for a dataset transfer property list
getBtreeRatios	Gets B-tree split ratios for a dataset transfer property list
setVlenMemManager	Sets the memory manager for variable-length datatype allocation in H5Dread and H5Dvlen_reclaim
getVlenMemManager	Gets the memory manager for variable-length datatype allocation in H5Dread and H5Tvlen_reclaim
~DSetMemXferPropList	Virtual destructor

3. Examples

The following examples show the application of some of the available functionality:

- create.cpp: writes a dataset to an HDF5 file. Specific functions used include:
  - ◊ constructors of H5File, DataSpace, and IntType
  - ◊ H5File::createDataSet to create a new dataset in this file
  - ◊ DataSet::write
  - ◊ DataType::setOrder
  - ◊ exception handlings

- readdata.cpp: obtains dataset information from an HDF5 file and reads selected data from the file. Specific functions, that are not in the previous example, include:
  - ◊ H5File::openDataSet to open an existing dataset that belongs to this file
  - ◊ DataSet::getTypeClass to get the type class identifier of the datatype of this dataset to determine what type is to be expected, in this case, it is H5T_INTEGER, i.e. the datatype is an integer
  - ◊ constructor an empty IntType instance after knowing what the expecting type is; this IntType object is passed into the subsequent function
  - ◊ DataSet::getType to retrieve the dataset's datatype which is an integer
  - ◊ IntType::getOrder
  - ◊ IntType::getSize
  - ◊ DataSet::getSpace
  - ◊ DataSpace::getSimpleExtentNdims
  - ◊ DataSpace::getSimpleExtentDims
  - ◊ DataSpace::selectHyperslab
  - ◊ DataSet::read

- writedata.cpp: writes selected data to an HDF5 file. Specific functions that are not in the previous examples include:
  - ◊ DataSpace::selectNone
  - ◊ DataSpace::selectElements

- compound.cpp: creates a compound datatype, write an array, which has the compound datatype to the file, and read back fields' subsets. Specific functions that are not in the previous examples include:
  - ◊ constructor CompType
  - ◊ CompType::insertMember to insert some members into the compound datatype
  - ◊ CompType::getMemberClass to get the type class identifier to determine what type is to be expected, in this case, it is H5T_FLOAT, i.e. the member datatype is floating-point
  - ◊ constructor an empty FloatType instance after knowing what the expecting type is; this FloatType object is passed into the subsequent function.
  - ◊ CompType::getMemberType to retrieve the specific datatype, FloatType
  - ◊ FloatType::getNorm to get the mantissa normalization of the floating-point datatype

- extend_ds.cpp: works with extendible dataset. Specific functions that are not in the previous examples include:
  - ◊ constructor of PropList
  - ◊ DsetCreatPropList::setChunk
  - ◊ DataSet::extend

- chunks.cpp: reads data from a chunked dataset. Specific functions that are not in the previous examples include:
  - ◊ DataSet::getSpace to get the dataspace in the file of this dataset
  - ◊ DataSet::getCreatePlist to get the dataset creation property list
  - ◊ DsetCreatPropList::getLayout
  - ◊ DsetCreatPropList::getChunk

- h5group.cpp: creates and . Specific functions that are not in the previous examples include:
  - ◊ H5File::createGroup to get a group in the file
  - ◊ constructor of DsetCreatPropList
  - ◊ DsetCreatPropList::setDeflate
  - ◊ H5File::openGroup to open a group in the file
  - ◊ Group::openDataSet to open a dataset in the group
  - ◊ H5File::link to create a hard link to a group
  - ◊ H5File::unlink to remove the hard link

`hdfhelp@ncsa.uiuc.edu`
Last modified: 19 December 2000