HDF5 C++ API Design Specification
Table of Contents
First Draft, September 2004

HDF5 C++ API Design Specification

First Draft, September 2004

1. Background

This document is intended to specify the details that are followed to implement the C++ API of the HDF5 library. For users, who need to obtain information on how to use the API, the *HDF5 C++ API Reference Manual* is recommended and can be found at `http://hdf.ncsa.uiuc.edu/HDF5/doc/cpplus_RM/`.

It is assumed that the reader of this document has knowledge of the HDF5 file format and its components. The HDF5 C++ API provides C++ wrappers for the HDF5 C library. Because the HDF5 library maps very well to the object oriented design approach, classes in the C++ API can closely represent the interfaces of the HDF5 APIs. Table 1 shows how the HDF5 C API are mapped into the C++ API classes.

Table 1. HDF5 C APIs mapped into C++ classes

HDF5 C APIs	C++ Classes
H5A - Attribute Interface	Attribute
H5D - Datasets Interface	DataSet
H5E - Error Interface	Exception
H5F - File Interface	H5File
H5G - Group Interface	Group
H5I - Identifier Interface	IdComponent
H5P - Property List Interface	PropList
H5S - Dataspace Interface	DataSpace
H5T - Datatype Interface	DataType
H5Z - Filters and Compression Interface	FilterCompress (not done)

The document starts with the Description section that describes the structure of the API. Following it are the Classes Description, Operations Description, and Data Dictionary sections describing the classes, their operations, and members. Analysis and Design Rationale explains design decisions. The document ends with two sections listing the current status and future work of the API.

2. Description of the HDF5 C++ API

This section describes the structure of the HDF5 C++ API. It explains the details of how the HDF5 interfaces are mapped into the C++ API classes. All classes are included in a namespace called `H5`.

Table 1 above lists the initial classes that are mapped from the HDF5 C APIs. Beside these initial classes, several classes are derived. Table 2 lists all the implemented classes and their purpose. Figure 1, Figure 2, Figure 3, Figure 4, and Figure 5 provide the diagrams showing the hierarchical relationship between the classes.

Table 2. All classes in the HDF5 C++ API

Class	Description
AbstractDs	base class for commonalities of `DataSet` and `Attribute`
AtomType	is a `DataType`; base class for commonalties of HDF5 predefined provides functions that access an enumeration datatype
Attribute	provides functions that access an attribute
CommonFG	is a base class for commonalities of `H5File` and `Group`
CompType	is a `DataType`; provides functions that access a compound datatype
DataSet	provides functions that access a dataset
DataSpace	provides functions that access the HDF5 dataspace
DataType	is an `H5Object`; provides functions that access a general datatype, which can be an enumeration datatype, compound datatype, or atomic datatype
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
EnumType	is a `DataType`; provides functions that access an enumeration datatype
Exception	provides the mechanism for handling errors returned by the C HDF5 library; it has several subclasses for specific exceptions as listed below
FileAccPropList	is a `PropList`; provides accesses to a file access property list
FileCreatPropList	is a `PropList`; provides accesses to a file creation property list
FloatType	is an `AtomType`; provides functions that access an floating-point datatype
Group	is an `H5Object`; provides functions that access HDF5 groups
H5File	provides functions that access an HDF5 file
H5Library	provides general-purpose library functions
H5Object	base class for commonalties of all HDF5 objects which include groups, datasets, datatypes, and attributes
IdComponent	is a base class that manage HDF5 object identifier
IntType	is an `AtomType`; provides functions that access an integer datatype
PredType	is an `AtomType`; provides the constant `PredType` objects for all the predefined datatype provided by the HDF5 library
PropList	provides common accesses to the property lists
RefCounter	provides reference counting mechanism
StrType	is an `AtomType`; provides functions that access an string datatype
AttributeIException	`Attribute` exception
DataSetIException	`DataSet` exception
DataSpaceIException	`DataSpace` exception
DataTypeIException	`DataType` exception
FileIException	`H5File` exception
GroupIException	`Group` exception
IdComponentException	`IdComponent` exception

LibraryIException	`H5Library` exception
PropListIException	`PropList` exception

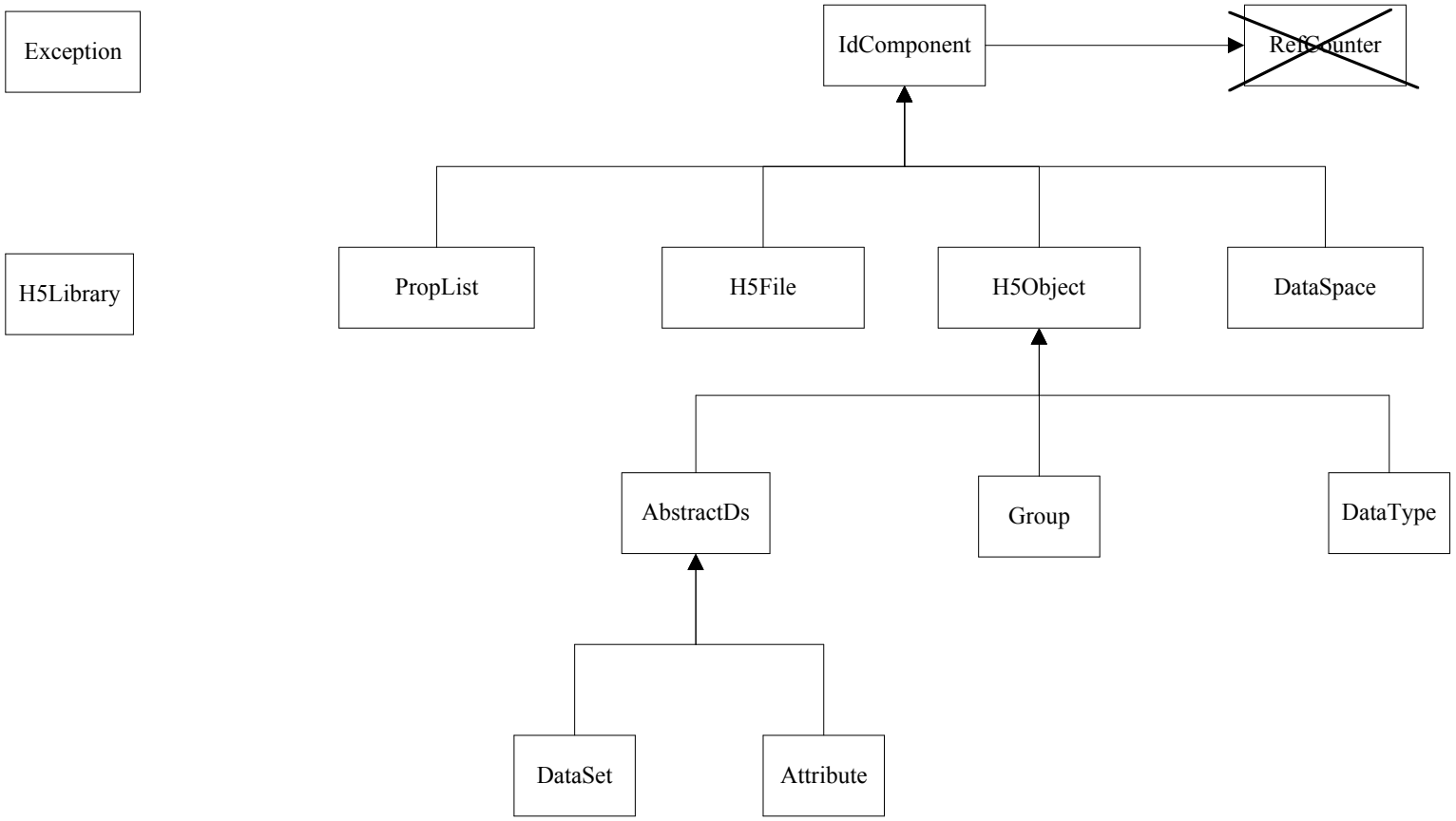

Figure 1. Class hierarchy of HDF5 C++ API

The figure shows that all the classes in the API, except `H5Library` and `Exception`, inherit from `IdComponent`. The HDF5 objects that are represented by the `IdComponent`'s subclasses are identified by HDF5 object identifiers (h5id's.) These identifiers are defined and manipulated by the HDF5 library. Class `IdComponent` relieves the C++ API users from the concern about properly managing the identifiers of any HDF5 objects, such as properly closing identified objects.

`H5Library` is a stand-alone class to provide accesses to the library as a whole. Class `Exception` has several subclasses for specific exceptions (Figure 2.) The Property List Interface provides several kinds of properties, which are then mapped into the derived classes of `PropList` (Figure 3.) Similarly, the `Datatype` Interface has different types and they are mapped into subclasses of `DataType` (Figure 4.) The two classes `H5File` and `Group` also inherit from another base class, `CommonFG`, because of their commonality that does not exist in other subclasses of `IdComponent` (Figure 5.)

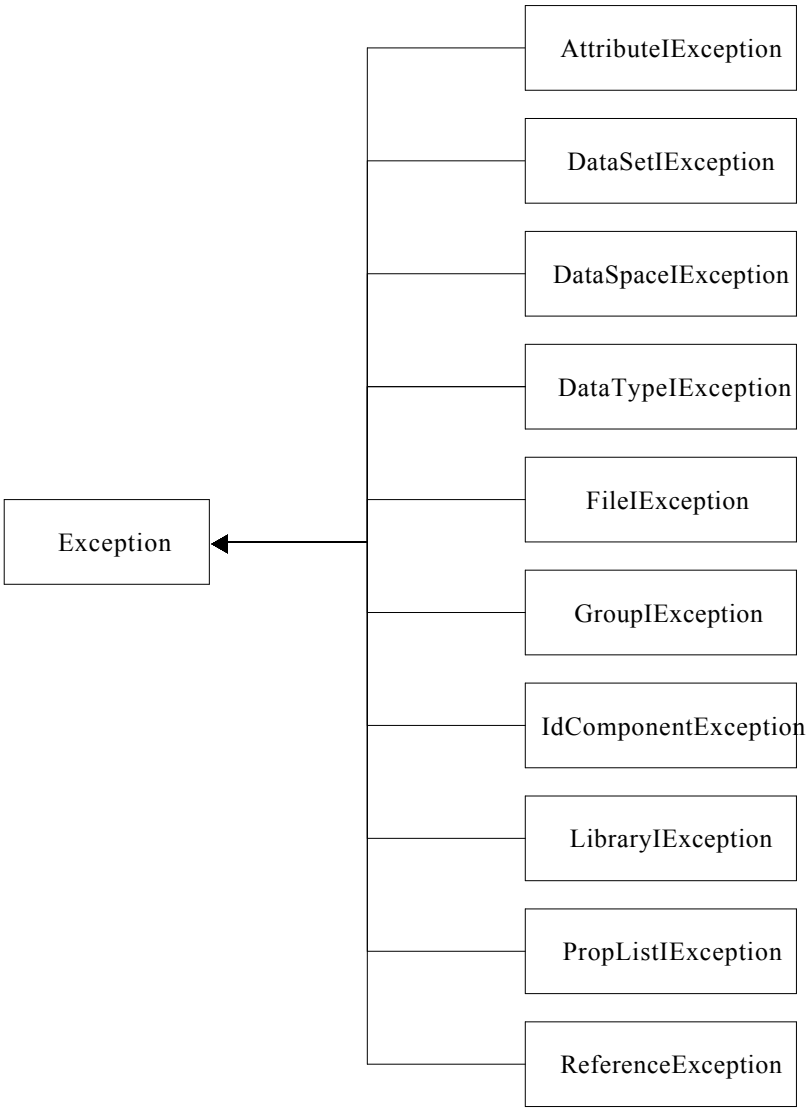

Figure 2 Exception hierarchy

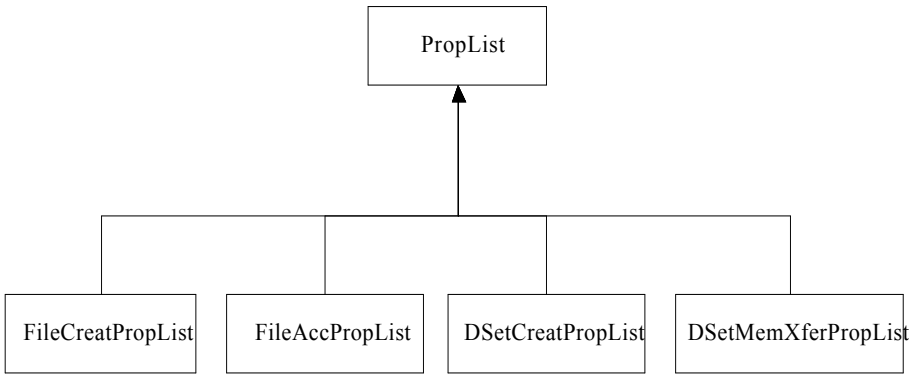

Figure 3 PropList hierarchy

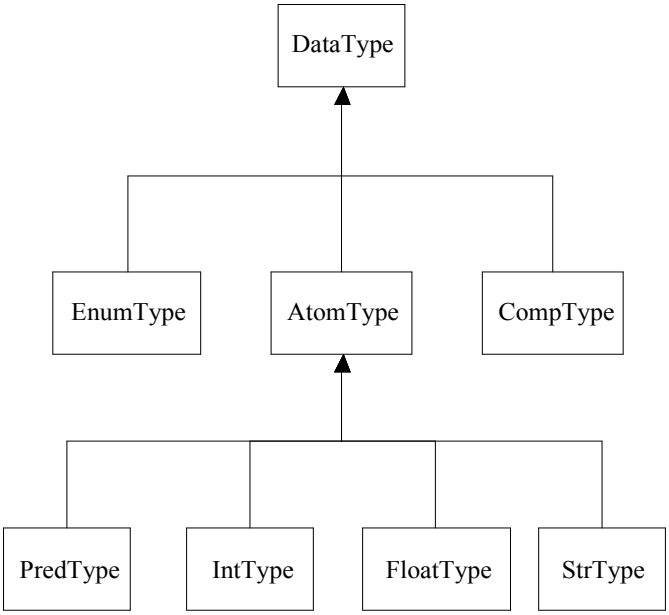

Figure 4 DataType hierarchy

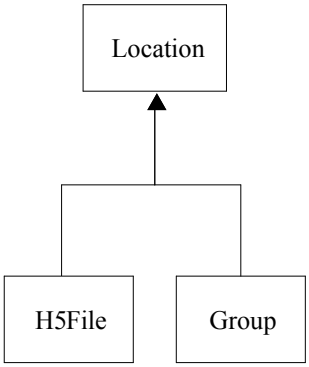

Figure 5 Location hierarchy

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

3. Classes Description

This section describes the classes that form the C++ API of HDF5. Each subsection below lists a class and the name of its members.

3.1. 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 publicly inherits from `H5Object`. Its members are listed in the table below and described in Section 4.1.

Member Function	Purpose
getCompType	Returns the compound datatype of this abstract dataset which can be a dataset or an attribute.
getDataType	Returns the generic datatype of this abstract dataset, which can be a dataset or an attribute.
getEnumType	Returns the enumeration datatype of this abstract dataset which can be a dataset or an attribute.
getFloatType	Returns the floating-point datatype of this abstract dataset, which can be a dataset or an attribute.
getIntType	Returns the integer datatype of this abstract dataset which can be a dataset or an attribute.
getSpace	
getStrType	Returns the string datatype of this abstract dataset which can be a dataset or an attribute.
getTypeClass	Returns the class of the datatype that is used by this object, which can be a dataset or an attribute.
AbstractDs	Copy constructor: makes a copy of the original AbstractDs object.
~AbstractDs	Noop destructor.

3.2. AtomType

This class inherits from `DataType` and, in addition, provides common services to access HDF5's predefined types, integer type, floating-point type, and string type. Its members are listed in the table below and described in Section 4.2.

Member Function	Purpose
getOrder	Returns the byte order of an atomic datatype.
setOrder	Sets the byte ordering 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	Retrieves the padding type of the least and most-significant bit padding.
setPad	Sets the least and most-significant bits padding types.

getPrecision	Returns the precision of an atomic datatype.
setPrecision	Sets the precision of an atomic datatype.
setSize	Sets the total size for an atomic datatype.
AtomType	Copy constructor: makes a copy of the original AtomType object.
~AtomType	Noop destructor.

3.3. Attribute

This class inherits from `AbstractDs` and provides additional accesses specific to an HDF5 object's attribute. Its members are listed in the table below and described in Section 4.3.

Member Function	**Purpose**
getName	Gets the name of this attribute, returning its length.
getName	This is an overloaded member function, provided for convenience. It differs from the above function in that it returns the attribute's name, not the length.
getName	This is an overloaded member function, provided for convenience. It differs from the above functions in that it doesn't take any arguments and returns the attribute's name.
getSpace	Gets a copy of the dataspace for this attribute.
read	Reads data from this attribute.
read	This is an overloaded member function, provided for convenience. It reads a std::string from this attribute.
write	Writes data to this attribute.
write	This is an overloaded member function, provided for convenience. It writes a std::string to this attribute.
Attribute	Creates an Attribute object using the id of an existing attribute.
Attribute	Copy constructor: makes a copy of the original Attribute object.
~Attribute	Properly terminates access to this attribute.

Notes on implementation:

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

3.4. CommonFG

`CommonFG` means commonality between file and group. This class is a protocol class. It has the common services that are provided by both `H5File` and `Group`. The file or group in the context of this class is referred to as 'location'. Member functions of this class are described in Section 4.4.

Member Function	Purpose
createGroup	Creates a new group at this location which can be a file or another group.
createGroup	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for name.

openGroup	Opens an existing group in a location which can be a file or another group.
openGroup	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for name.
createDataSet	Creates a new dataset at this location.
createDataSet	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for name.
openDataSet	Opens an existing dataset at this location.
openDataSet	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for name.
getComment	Retrieves comment for the specified object.
getComment	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for name.
setComment	Sets or resets the comment for an object specified by its name.
void	setComment (const string &name, const string &comment) const
	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for name and comment.
getLinkval	Returns the name of the object that the symbolic link points to.
string	getLinkval (const string &name, size_t size) const
	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for name.
getNumObjs	Returns the number of objects in this group.
getObjinfo	Returns information about an object.
getObjinfo	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for name.
getObjnameByIdx	Retrieves the name of an object in this group, given the object's index.
getObjTypeByIdx	Returns the type of an object in this group, given the object's index.
H5G_obj_t	getObjTypeByIdx (hsize_t idx, string &type_name) const
	This is an overloaded member function, provided for convenience. It differs from the above function because it also provides the returned object type in text.
iterateElems	Iterates a user's function over the entries of a group.
int	iterateElems (const string &name, int *idx, H5G_iterate_t op, void *op_data)
	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for name.
link	Creates a link of the specified type from new_name to curr_name.
void	link (H5G_link_t link_type, const string &curr_name, const string &new_name) const
	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for curr_name and new_name.
unlink	Removes the specified name at this location.
void	unlink (const string &name) const
	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for name.
mount	Mounts the file child onto this group.
void	mount (const string &name, H5File &child, PropList &plist) const
	This is an overloaded member function, provided for convenience. It differs from the

	above function in that it takes an std::string for name.
unmount	Unmounts the specified file.
void	unmount (const string &name) const
	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for name.
move	Renames an object at this location.
void	move (const string &src, const string &dst) const
	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for src and dst.
openDataType	Opens the named generic datatype at this location.
DataType	openDataType (const string &name) const
	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for name.
openEnumType	Opens the named enumeration datatype at this location.
EnumType	openEnumType (const string &name) const
	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for name.
openCompType	Opens the named compound datatype at this location.
CompType	openCompType (const string &name) const
	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for name.
openIntType	Opens the named integer datatype at this location.
IntType	openIntType (const string &name) const
	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for name.
openFloatType	Opens the named floating-point datatype at this location.
openFloatType	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for name.
openStrType	Opens the named string datatype at this location.
openStrType	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes an std::string for name.
virtual hid_t	getLocId () const =0
throwException	For subclasses, H5File and Group, to throw appropriate exception.
CommonFG	Default constructor.
~CommonFG	Noop destructor.

3.5. CompType

This class inherits from `DataType` and provides additional accesses to a compound datatype. Its members are listed in the table below and described in Section 4.5.

Member Function	Purpose
CompType	Creates an empty compound datatype given a size, in bytes.

CompType	Gets the compound datatype of the specified dataset.
getMemberClass	Gets the type class of the specified member.
getMemberDims	
getMemberIndex	Returns the index of a member in this compound datatype.
int	getMemberIndex (const string &name) const
getMemberOffset	Returns the byte offset of the beginning of a member with respect to the beginning of the compound data type datum.
getMemberName	Returns the name of a member in this compound datatype.
getMemberCompType	Returns the compound datatype of the specified member in this compound datatype.
getMemberDataType	Returns the generic datatype of the specified member in this compound datatype.
getMemberEnumType	Returns the enumeration datatype of the specified member in this compound datatype.
getMemberIntType	Returns the integer datatype of the specified member in this compound datatype.
getMemberFloatType	Returns the floating-point datatype of the specified member in this compound datatype.
getMemberStrType	Returns the string datatype of the specified member in this compound datatype.
getNmembers	Returns the number of members in this compound datatype.
insertMember	Inserts a new member to this compound datatype.
pack	Recursively removes padding from within a compound datatype.
CompType	Default constructor: Creates a stub compound datatype.
CompType	Creates a CompType object using the id of an existing datatype.
CompType	Copy constructor: makes copy of the original CompType object.
~CompType	Properly terminates access to this compound datatype.

3.6. DataSet

This class inherits from `AbstractDs` and provides additional accesses specific to an HDF5 dataset. Its members are listed in the table below and described in Section 4.6.

Member Function	Purpose
extend	Extends a dataset with unlimited dimension.
fillMemBuf	Fills a selection in memory with a value.
void	fillMemBuf (void *buf, DataType &buf_type, DataSpace &space)
	This is an overloaded member function, provided for convenience. It differs from the above function in that it only takes the the last three arguments.
getCreatePlist	Gets the dataset creation property list.
getOffset	Returns the address of this dataset in the file.
getSpace	Gets a copy of the dataspace of this dataset.
getSpaceStatus	Determines whether space has been allocated for a dataset.

getStorageSize	Returns the amount of storage required for a dataset.
getVlenBufSize	Returns the number of bytes required to store VL data.
vlenReclaim	Reclaims VL datatype memory buffers.
read	Reads raw data from the specified dataset.
void	read (string &buf, const DataType &mem_type, const DataSpace &mem_space=DataSpace::ALL, const DataSpace &file_space=DataSpace::ALL, const DSetMemXferPropList &xfer_plist=DSetMemXferPropList::DEFAULT) const
	This is an overloaded member function, provided for convenience. It takes a reference to a std::string for the buffer.
write	Writes raw data from an application buffer to a dataset.
void	write (const string &buf, const DataType &mem_type, const DataSpace &mem_space=DataSpace::ALL, const DataSpace &file_space=DataSpace::ALL, const DSetMemXferPropList &xfer_plist=DSetMemXferPropList::DEFAULT) const
	This is an overloaded member function, provided for convenience. It takes a reference to a std::string for the buffer.
iterateElems	Iterates over all selected elements in a dataspace.
getObjType	Retrieves the type of object that an object reference points to.
getRegion	Retrieves a dataspace with the region pointed to selected.
Reference	Creates a reference to an HDF5 object or a dataset region.
void *	Reference (const char *name) const
	This is an overloaded function, provided for your convenience. It differs from the above function in that it only creates a reference to an HDF5 object, not to a dataset region.
DataSet	Creates an DataSet object using the id of an existing dataset.
DataSet	Default constructor: creates a stub DataSet.
DataSet	Copy constructor: makes a copy of the original DataSet object.
~DataSet	Properly terminates access to this dataset.

3.7. DataSpace

This class inherits from `IdComponent` and provides accesses to an HDF5 dataspace. Its members are listed in the table below and described in Section 4.7.

Member Function	Purpose
DataSpace	Creates a new dataspace given a dataspace type.
DataSpace	Creates a new simple dataspace.
DataSpace &	(const DataSpace &rhs)
operator=	Assignment operator.
copy	Makes a copy of an existing dataspace.
extentCopy	Copies the extent of a dataspace.
getSelectBounds	Gets the bounding box containing the current selection.
getSelectElemNpoints	Returns the number of element points in the current selection.

getSelectElemPointlist	Gets the list of element points currently selected.
getSelectHyperBlocklist	Gets the list of hyperslab blocks currently selected.
getSelectHyperNblocks	Returns number of hyperslab blocks.
getSelectNpoints	Returns the number of elements in a dataspace selection.
getSimpleExtentDims	Retrieves dataspace dimension size and maximum size.
getSimpleExtentNdims	Returns the dimensionality of a dataspace.
getSimpleExtentNpoints	Returns the number of elements in a dataspace.
getSimpleExtentType	Returns the current class of a dataspace.
isSimple	Determines whether this dataspace is a simple dataspace.
offsetSimple	Sets the offset of this simple dataspace.
selectAll	Selects the entire dataspace.
selectElements	Selects array elements to be included in the selection for this dataspace.
selectHyperslab	Selects a hyperslab region to add to the current selected region.
selectNone	Resets the selection region to include no elements.
selectValid	Verifies that the selection is within the extent of the dataspace.
setExtentNone	Removes the extent from a dataspace.
setExtentSimple	Sets or resets the size of an existing dataspace.
DataSpace	Creates a DataSpace object using the id of an existing dataspace.
DataSpace	Copy constructor: makes a copy of the original DataSpace object.
~DataSpace	Properly terminates access to this dataspace.

3.8. DataType

This class inherits from `H5Object` and provides the additional accesses to an HDF5 generic datatype. Its members are listed in the table below and described in Section 4.8. Several subclasses are derived from `DataType`.

Member Function	Purpose
DataType	Creates a object given its class and size.
DataType	Copy constructor: makes a copy of the original DataType object.
copy	Copies an existing datatype to this datatype object.
getClass	Returns the datatype class identifier.
void	commit (CommonFG &loc, const string &name) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in the type of the argument name.
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 a conversion from this datatype to the specified datatype, dest.

convert	Converts data from this datatype to the specified datatypes.
operator=	Assignment operator.
operator==	Compares this DataType against the given one to determines whether the two objects refer to the same actual datatype.
lock	Locks a datatype, making it read-only and non-destructible.
getSize	Returns the size of a datatype.
getSuper	Returns the base datatype from which a datatype is derived.
void	registerFunc (H5T_pers_t pers, const string &name, const DataType &dest, H5T_conv_t func) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in the type of the argument name.
registerFunc	Registers the specified conversion function.
void	unregister (H5T_pers_t pers, const string &name, const DataType &dest, H5T_conv_t func) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in the type of the argument name.
unregister	Removes a conversion function from all conversion paths.
void	setTag (const string &tag) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in the type of the argument name.
setTag	Tags an opaque datatype.
getTag	Gets the tag associated with an opaque datatype.
detectClass	Checks whether a datatype contains (or is) a certain type of datatype.
isVariableStr	Check whether this datatype is a variable-length string.
void *	Reference (const char *name) const
	This is an overloaded function, provided for your convenience. It differs from the above function in that it only creates a reference to an HDF5 object, not to a dataset region.
Reference	Creates a reference to an HDF5 object or a dataset region.
getObjType	Retrieves the type of object that an object reference points to.
getRegion	Retrieves a dataspace with the region pointed to selected.
DataType	Creates a datatype using an existing datatype's id.
DataType	Default constructor: Creates a stub datatype.
~DataType	Properly terminates access to this datatype.

3.9. DSetCreatPropList

This class inherits from class `PropList` and provides accesses to the HDF5 dataset creation property lists. Its members are listed in the table below and described in Section 4.9.

Member Function	Purpose
allFiltersAvail	Queries whether all the filters set in this property list are available currently.
getAllocTime	Get space allocation time for this property.

setAllocTime	Sets space allocation time for dataset during creation.
getChunk	Retrieves the size of the chunks used to store a chunked layout dataset.
setChunk	Sets the size of the chunks used to store a chunked layout dataset.
getExternal	Returns information about an external file.
getExternalCount	Returns the number of external files for a dataset.
getFillTime	Gets fill value writing time.
setFillTime	Sets fill value writing time for dataset.
getFillValue	Retrieves a dataset fill value.
setFillValue	Sets a dataset fill value.
getFilter	Returns information about a filter in a pipeline.
getFilterById	Returns information about a filter in a pipeline given the filter id.
getLayout	Retrieves the layout type of this property list.
setLayout	Sets the type of storage used store the raw data for a dataset.
getNfilters	Returns the number of filters in the pipeline.
isFillValueDefined	Check if fill value has been defined for this property.
modifyFilter	Modifies the specified filter.
removeFilter	Removes one or more filters.
setDeflate	Sets compression method and compression level.
setExternal	Adds an external file to the list of external files.
setFilter	Adds a filter to the filter pipeline.
setFletcher32	Sets Fletcher32 checksum of EDC for this property list.
setShuffle	Sets method of the shuffle filter.
DSetCreatPropList	Default constructor: creates a stub dataset creation property list.
DSetCreatPropList	Copy constructor: makes a copy of the original DSetCreatPropList object.
DSetCreatPropList	Creates a DSetCreatPropList object using the id of an existing dataset creation property list.
~DSetCreatPropList	Noop destructor.

3.10. DSetMemXferPropList

This class inherits from class `PropList` and provides accesses to the HDF5 data set memory and transfer property lists. Its members are listed in the table below and described in Section 4.10.

Member Function	Purpose
setBtreeRatios	Sets B-tree split ratios for a dataset transfer property list.
getBtreeRatios	Gets B-tree split ratios for a dataset transfer property list.
setBuffer	Sets type conversion and background buffers.
getBuffer	Reads buffer settings.

setEDCCheck	Enables or disables error-detecting for a dataset reading process.
getEDCCheck	Determines whether error-detection is enabled for dataset reads.
setHyperVectorSize	Sets number of I/O vectors to be read/written in hyperslab I/O.
getHyperVectorSize	Returns the number of I/O vectors to be read/written in hyperslab I/O.
setMulti	Sets the data transfer property list for the multi-file driver.
getMulti	Returns multi-file data transfer property list information.
setPreserve	Sets the dataset transfer property list status to true or false.
getPreserve	Checks status of the dataset transfer property list.
setSmallDataBlockSize	Sets the size of a contiguous block reserved for small data.
getSmallDataBlockSize	Returns the current small data block size setting.
setVlenMemManager	Sets the memory manager for variable-length datatype allocation.
setVlenMemManager	Sets the memory manager for variable-length datatype allocation - system malloc and free will be used.
getVlenMemManager	Gets the memory manager for variable-length datatype allocation.
DSetMemXferPropList	Default constructor: creates a stub dataset memory and transfer property list object.
DSetMemXferPropList	Copy constructor: makes a copy of the original DSetMemXferPropList object.
DSetMemXferPropList	Creates a DSetMemXferPropList object using the id of an existing DSetMemXferPropList.
~DSetMemXferPropList	Noop destructor.

3.11. EnumType

This class inherits from `DataType` and provides additional accesses to an enumeration datatype. Its members are listed in the table below and described in Section 4.11.

Member Function	Purpose
EnumType	Creates an empty enumeration datatype given a size, in bytes.
EnumType	Gets the enum datatype of the specified dataset.
EnumType	Creates a new enum datatype based on an integer datatype.
getNmembers	Returns the number of members in this enumeration datatype.
getMemberIndex	Returns the index of a member in this enumeration datatype.
getMemberIndex	This is an overloaded member function, provided for convenience. It differs from the above function only in the type of argument name.
getMemberValue	Retrieves the value of a member in this enumeration datatype, given the member's index.
insert	Inserts a new member to this enumeration datatype.
	insert (const string &name, void *value) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in the type of argument name.
nameOf	Returns the symbol name corresponding to a specified member of this enumeration

	datatype.
valueOf	Retrieves the value corresponding to a member of this enumeration datatype, given the member's name.
	valueOf (const string &name, void *value) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in the type of argument name.
EnumType	Default constructor: Creates a stub datatype.
EnumType	Creates an EnumType object using the id of an existing datatype.
EnumType	Copy constructor: makes a copy of the original EnumType object.
~EnumType	Properly terminates access to this enum datatype.

3.12. Exception

This class provides exception handlings. It has two data members, `detail_message` and `func_name`. 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 members of `Exception` and their details are in Section 4.12.

Data Member	Description
detail_message **func_name**	string; stores a descriptive message of the exception string; stores the name of the function from which the exception is thrown
Member Function	Purpose
Exception	Creates an exception with the name of the function, in which the failure occurs, and an optional detailed message.
clearErrorStack	Clears the error stack for the current thread.
dontPrint	Turns off the automatic error printing from the C library.
getAutoPrint	Retrieves the current settings for the automatic error stack traversal function and its data.

setAutoPrint	Turns on the automatic error printing.
getCDetailMsg	Returns the detailed message set at the time the exception is thrown.
getCFuncName	Returns the name of the function, where the exception is thrown.
getDetailMsg	Returns the detailed message set at the time the exception is thrown.
getFuncName	Returns the name of the function, where the exception is thrown.
getMajorString	Returns a text string that describes the error specified by a major error number.
getMinorString	Returns a text string that describes the error specified by a minor error number.
printError	Prints the error stack in a default manner.
walkErrorStack	Walks the error stack for the current thread, calling the specified function
Exception	Default constructor.
Exception	Copy constructor: makes a copy of the original Exception object.
~Exception	Noop destructor.

3.13. FileAccPropList

This class inherits from class `PropList` and provides accesses to the HDF5 file access property lists. Its members are listed in the table below and described in Section 4.13.

Member Function	Purpose
setStdio	Modifies this property list to use the H5FD_STDIO driver.
setDriver	Set file driver for this property list.
getDriver	Return the ID of the low-level file driver.
setFamilyOffset	Sets offset for family driver.
getFamilyOffset	Get offset for family driver.
setSec2	Modifies this file access property list to use the sec2 driver.
setCore	Modifies this file access property list to use the H5FD_CORE driver.
getCore	Queries core file driver properties.
setFamily	Sets this file access property list to use the family driver.
getFamily	Returns information about the family file access property list.
	getFamily (hsize_t &memb_size) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in what arguments it accepts and its return value.
setSplit	Emulates the old split file driver, which stored meta data in one file and raw data in another file.
	setSplit (FileAccPropList &meta_plist, FileAccPropList &raw_plist, const string &meta_ext, const string &raw_ext) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in what arguments it accepts.
setSieveBufSize	Sets the maximum size of the data sieve buffer.
getSieveBufSize	Returns the current settings for the data sieve buffer size property from this property list.

setMetaBlockSize	Sets the minimum size of metadata block allocations.
getMetaBlockSize	Returns the current metadata block size setting.
setLog	Modifies this file access property list to use the logging driver.
	setLog (const string &logfile, unsigned flags, size_t buf_size) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in what arguments it accepts.
setAlignment	Sets the alignment properties of this property list.
getAlignment	Returns the current settings for alignment properties from this property list.
setMultiType	Sets data type for MULTI driver.
getMultiType	Returns the data type property for MULTI driver.
setCache	Sets the meta data cache and raw data chunk cache parameters. Sets the meta data cache and raw data chunk cache parameters.
getCache	Queries the meta data cache and raw data chunk cache parameters.
setFcloseDegree	Sets the degree for the file close behavior.
getFcloseDegree	Returns the degree for the file close behavior.
setGcReferences	Sets garbage collecting references flag.
getGcReferences	Returns the garbage collecting references setting.
FileAccPropList	Default constructor: creates a file access property list.
FileAccPropList	Copy constructor: makes a copy of the original FileAccPropList. FileAccPropList object.
FileAccPropList	Creates a file access property list using the id of an existing one.
~FileAccPropList	Noop destructor.

3.14. FileCreatPropList

This class inherits from class `PropList` and provides accesses to the HDF5 file creation property lists. Its members are listed in the table below and described in Section 4.14.

Member Function	Purpose
FileCreatPropList	Default constructor: creates a file creation property list.
getIstorek	Returns the 1/2 rank of an indexed storage B-tree.
setIstorek	Sets the size of the parameter used to control the B-trees for indexing chunked datasets.
getSizes	Retrieves the size of the offsets and lengths used in an HDF5 file.
setSizes	Sets the byte size of the offsets and lengths used to address objects in an HDF5 file.
getSymk	Retrieves the size of the symbol table B-tree 1/2 rank and the symbol table leaf node 1/2 size.
setSymk	Sets the size of parameters used to control the symbol table nodes.
getUserblock	Returns the user block size of this file creation property list.
setUserblock	Sets the user block size field of this file creation property list.
getVersion	Retrieves version information for various parts of a file.

FileCreatPropList	Copy constructor: makes a copy of the original FileCreatPropList object.
FileCreatPropList	Creates a file creation property list using the id of an existing one.
~FileCreatPropList	Noop destructor.

3.15. FloatType

This class inherits from `AtomType` and provides additional accesses to the user-defined floating-point datatype. Its members are listed in the table below and described in Section 4.15.

Member Function	Purpose
FloatType	Creates a floating-point datatype using a predefined type.
FloatType	Gets the floating-point datatype of the specified dataset.
getEbias	Retrieves the exponent bias of a floating-point type.
setEbias	Sets the exponent bias of a floating-point type.
getFields	Retrieves floating point datatype bit field information.
setFields	Sets locations and sizes of floating point bit fields.
getInpad	Retrieves the internal padding type for unused bits in this floating-point datatypes.
setInpad	Fills unused internal floating point bits.
getNorm	Retrieves mantissa normalization of a floating-point datatype.
setNorm	Sets the mantissa normalization of a floating-point datatype.
FloatType	Default constructor: Creates a stub floating-point datatype.
FloatType	Creates an FloatType object using the id of an existing datatype.
FloatType	Copy constructor: makes a copy of the original FloatType object.
~FloatType	Noop destructor.

3.16. Group

Group inherits from `H5Object` and `CommonFG`. In addition, it provides some functionality that is specific to an HDF5 group. Its members are listed in the table below and described in Section 4.16.

Member Function	Purpose
getObjType	Retrieves the type of object that an object reference points to.
getRegion	Retrieves a dataspace with the region pointed to selected.
Reference	Creates a reference to an HDF5 object or a dataset region.
	Reference (const char *name) const
	This is an overloaded function, provided for your convenience. It differs from the above function in that it only creates a reference to an HDF5 object, not to a dataset region.
throwException	Throws H5::GroupIException.
getLocId	Returns the id of this group.

Group	Default constructor: creates a stub Group.
Group	Copy constructor: makes a copy of the original Group object.
Group	Creates a Group object using the id of an existing group.
~Group	Properly terminates access to this group.

3.17. H5File

This class inherits from `IdComponent` and `CommonFG`. In addition, it provides some accesses to the HDF5 files and these functions are listed in the following table and described in Section 4.17.

Member Function	Purpose
H5File	Creates or opens an HDF5 file depending on the parameter flags.
	H5File (const string &name, unsigned int flags, const FileCreatPropList &create_plist=FileCreatPropList::DEFAULT, const FileAccPropList &access_plist=FileAccPropList::DEFAULT)
	This is another overloaded constructor. It differs from the above constructor only in the type of the name argument.
getAccessPlist	Returns the access property list of this file.
getCreatePlist	Returns the creation property list of this file.
getFileName	Gets the name of this file.
getFileSize	Returns the file size of the HDF5 file.
getFreeSpace	Returns the amount of free space in the file.
getObjCount	Returns the number of opened object IDs (files, datasets, groups and datatypes) in the same file.
	getObjCount () const
	This is an overloaded member function, provided for convenience. It takes no parameter and returns the object count of all object types.
getObjIDs	Retrieves a list of opened object IDs (files, datasets, groups and datatypes) in the same file.
getObjType	Retrieves the type of object that an object reference points to.
getRegion	Retrieves a dataspace with the region pointed to selected.
getVFDHandle	Returns the pointer to the file handle of the low-level file driver.
	getVFDHandle (void **file_handle) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in what arguments it accepts.
reOpen	Reopens this file.
reopen	Reopens this file.
Reference	Creates a reference to an Hdf5 object or a dataset region.
	Reference (const char *name) const
	This is an overloaded function, provided for your convenience. It differs from the above function in that it only creates a reference to an HDF5 object, not to a dataset region.
throwException	Throws file exception - initially implemented for CommonFG.

getLocId	Returns the Id of the current HDF5 file.
H5File	Default constructor: creates a stub H5File object.
H5File	Copy constructor: makes a copy of the original H5File object.
~H5File	Properly terminates access to this file.

3.18. H5Library

This class provides accesses to the HDF5 library. It is independent with other classes in the API. All of its member functions are static. The function descriptions are in Section 4.18.

Member Functions	Purpose
H5Library	Default constructor - private
open	Initializes the HDF5 library.
close	Flushes all data to disk, closes files, and cleans up memory.
checkVersion	Verifies that the arguments match the version numbers compiled into the library.
dontAtExit	Instructs library not to install atexit cleanup routine.
garbageCollect	Walks through all the garbage collection routines for the library, which are supposed to free any unused memory they have allocated.
getLibVersion	Returns the HDF library release number.
setFreeListLimits	Sets limits on the different kinds of free lists

3.19. H5Object

This class inherits from `IdComponent` and provides some common accesses to an HDF5 object, which can be a group, a dataset, an attribute, or a named datatype. These functions are listed in the table below and described in Section 4.19.

Member Function	Purpose
createAttribute	Creates an attribute for a group, dataset, or named datatype.
	createAttribute (const string &name, const DataType &type, const DataSpace &space, const PropList &create_plist=PropList::DEFAULT) const
	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes a reference to an std::string for name.
openAttribute	Opens an attribute given its name.
	openAttribute (const string &name) const
	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes a reference to an std::string for name.
openAttribute	Opens an attribute given its index.
flush	Flushes all buffers associated with a file to disk.
getFileName	Gets the name of the file, in which this HDF5 object belongs.
getNumAttrs	Returns the number of attributes attached to this HDF5 object.
iterateAttrs	Iterates a user's function over all the attributes of an H5 object, which may be a group, dataset or named datatype.

removeAttr	Removes the named attribute from this object.
	removeAttr (const string &name) const
	This is an overloaded member function, provided for convenience. It differs from the above function in that it takes a reference to an std::string for name.
H5Object	Copy constructor: makes a copy of the original H5Object instance.
~H5Object	Noop destructor.

3.20. 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. However, in future versions, the reference counting will be handled by the C library, thus `RefCounter` will be removed. Section 4.20 describes the member functions in this class.

Data Member	Description
id	hid_t
ref_count	RefCounter* (this member will be removed)
Member Function	Purpose
incRefCount	Increments id reference counter.
decRefCount	Decrements id reference counter.
getCounter	Returns the reference counter to this identifier.
noReference	Determines whether this object has any references.
operator=	Assignment operator.
reset	Reset this object by deleting/resetting its reference counter.
setId	Sets the identifier of this object to a new value.
IdComponent	Creates an IdComponent object using the id of an existing object.
IdComponent	Copy constructor: makes a copy of the original IdComponent object.
getId	Returns the id of this object.
~IdComponent	Noop destructor.

3.21. IntType

This class inherits from `AtomType` and provides additional accesses to the user-defined integer datatype. Its members are listed in the table below and described in Section 4.21.

Member Function	Purpose
IntType	Creates a integer type using a predefined type.

IntType	Gets the integer datatype of the specified dataset.
getSign	Retrieves the sign type for an integer type.
setSign	Sets the sign property for an integer type.
IntType	Default constructor: Creates a stub integer datatype.
IntType	Creates an integer datatype using the id of an existing datatype.
IntType	Copy constructor: makes a copy of the original IntType object.
~IntType	Noop destructor.

3.22. PredType

This class contains the definition of objects that correspond to the predefined datatypes defined in the HDF5 library. They are listed in the Data Dictionary section. Its members are listed in the table below and described in Section 4.22.

Data Member	Purpose
predefined_types	Enumeration used to determine PredType id from HDF5 predefined types.
Member Function	Purpose
getId	Returns the id of a predefined datatype
operator=	Assignment operator
~PredType	Noop destructor
Protected:	
PredType	Default constructor
PredType	Copy constructor
PredType	Constructor with predefined type id

3.23. PropList

This class inherits from `IdComponent` and provides common services for the HDF5 file and data set property lists. Its member are listed in the table below and described in Section 4.23. Several subclasses are derived from `PropList`.

Member Function	Purpose
operator=	Assignment operator.
operator==	Compares this property list or class against the given list or class.
closeClass	Close a property list class.
copy	Makes a copy of an existing property list.
copyProp	Copies a property from one list or class to another.
	copyProp (PropList &dest, PropList &src, const string &name) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in what arguments it accepts.

getClass	Returns the class of this property list, i.e. H5P_FILE_CREATE...
getClassName	Return the name of a generic property list class.
getClassParent	Returns the parent class of a generic property class.
getNumProps	Returns the number of properties in this property list or class.
getProperty	Query the value of a property in a property list.
	getProperty (const char *name) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in what arguments it accepts.
	getProperty (const string &name, void *value) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in what arguments it accepts.
	getProperty (const string &name) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in what arguments it accepts.
setProperty	Set a property's value in a property list.
	setProperty (const char *name, const char *value) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in what arguments it accepts.
	setProperty (const char *name, string &strg) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in what arguments it accepts.
	setProperty (const string &name, void *value) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in what arguments it accepts.
	setProperty (const string &name, string &strg) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in what arguments it accepts.
getPropSize	Query the size of a property in a property list or class.
	getPropSize (const string &name) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in what arguments it accepts.
isAClass	Determines whether a property list is a certain class.
propExist	Query the existance of a property in a property object.
	propExist (const string &name) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in what arguments it accepts.
removeProp	Removes a property from a property list.
	removeProp (const string &name) const
	This is an overloaded member function, provided for convenience. It differs from the above function only in what arguments it accepts.
PropList	Default constructor: creates a stub property list object.
PropList	Creates a property list using the id of an existing property.
PropList	Copy constructor.

~PropList	Properly terminates access to this property list.

3.24. 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. Update: this class will be removed from the HDF5 C++ API in the next release because the HDF5 library has provided the reference counting mechanism at the library layer.

3.25. StrType

This class inherits from `AtomType` and provides additional accesses to the user-defined string datatype. Its members are listed in the table below and described in Section 4.25.

Member Function	Purpose
StrType	Creates a string datatype using a predefined type.
StrType	(const size_t &size)
StrType	Creates a string datatype with a specified length.
StrType	Gets the string datatype of the specified dataset.
getCset	Retrieves the character set type of this string datatype.
setCset	Sets character set to be used.
getStrpad	Retrieves the storage mechanism for of this string datatype.
setStrpad	Defines the storage mechanism for this string datatype.
StrType	Default constructor: Creates a stub string datatype.
StrType	Creates an StrType object using the id of an existing datatype.
StrType	Copy constructor: makes a copy of the original StrType object.
~StrType	Properly terminates access to this string datatype.

4. Operation Descriptions

This section describes all the member functions of the classes. Most of these functions are wrappers of a C API, which is specified in the line **C API**.

4.1. AbstractDs

4.1.1. AbstractDs default constructor

Prototype `AbstractDs() : H5Object()`

Description No operation

4.1.2. AbstractDs copy constructor

Prototype
```
AbstractDs(const AbstractDs& original) :
H5Object(original)
```

Description No operation

4.1.3. AbstractDs constructor using an existing id

Prototype `AbstractDs(const hid_t ds_id) : H5Object(ds_id)`
Description No operation

4.1.4. getSpace

Prototype `virtual DataSpace getSpace() const = 0`
Description This function is a pure virtual. DataSet and Attribute provide their own implementation.

4.1.5. getTypeClass

Prototype `H5T_class_t getTypeClass() const`
C API `H5Tget_class`
Description Invoke the private function `p_getType` to obtain the id of the datatype that is used by this dataset then call `H5Tget_class` to get the class of the datatype. Return the datatype class if it is not `H5T_NO_CLASS`.
Exception `DataTypeIException` if `H5Tget_class` returns `H5T_NO_CLASS`

4.1.6. getDataType

Prototype `DataType getDataType() const`
Description Invoke the private function `p_getType` to obtain the id of the datatype used by this dataset. Create and return the `DataType`.

4.1.7. getEnumType

Prototype `EnumType getEnumType() const`
Description Invoke the private function `p_getType` to obtain the id of the enumeration datatype used by this dataset. Create and return the `EnumType`.

4.1.8. getCompType

Prototype `CompType getCompType() const`
Description Invoke the private function `p_getType` to obtain the id of the compound datatype used by this dataset. Create and return the `CompType`.

4.1.9. getIntType

Prototype `IntType getIntType() const`
Description Invoke the private function `p_getType` to obtain the id of the integer datatype used by this dataset. Create and return the `IntType`.

4.1.10. getFloatType

Prototype	`FloatType getFloatType() const`
Description	Invoke the private function `p_getType` to obtain the id of the floating-point datatype used by this dataset. Create and return the `FloatType`.

4.1.11. getStrType

Prototype	`StrType getStrType() const`
Description	Invoke the private function `p_getType` to obtain the id of the string datatype used by this dataset. Create and return the `StrType`.

4.1.12. ~AbstractDs virtual destructor

Prototype	`virtual ~AbstractDs()`
Description	No operation

4.2. AtomType

4.2.1. AtomType default constructor

Prototype	`[protected] AtomType(): DataType()`
Description	No operation

4.2.2. AtomType copy constructor

Prototype	`AtomType(const AtomType& original) : DataType (original)`
Description	No operation

4.2.3. AtomType constructor using an existing id

Prototype	`[protected] AtomType(const hid_t existing_id) : DataType (existing_id)`
Description	No operation

4.2.4. setSize

Prototype	`void setSize(size_t size) const;`
C API	`H5Tset_size`
Exception	`DataTypeIException` if `H5Tset_size` returns a negative value

4.2.5. getOrder

Prototype	`H5T_order_t getOrder(string& order_string) const;`
C API	`H5Tget_order`
Description	Return the byte ordering obtained from `H5Tget_order`.
Exception	`DataTypeIException` if `H5Tget_order` returns `H5T_ORDER_ERROR` for type order.

4.2.6. setOrder

Prototype	`void setOrder(H5T_order_t order) const;`
C API	`H5Tset_order`
Exception	`DataTypeIException` if `H5Tset_order` returns a negative value

4.2.7. getPrecision

Prototype	`size_t getPrecision() const;`
C API	`H5Tget_precision`
Description	Return the number of significant bits obtained from `H5Tget_precision` if it is not 0.
Exception	`DataTypeIException` if `H5Tget_precision` returns 0.

4.2.8. setPrecision

Prototype	`void setPrecision(size_t precision) const;`
C API	`H5Tset_precision`
Exception	`DataTypeIException` if `H5Tset_precision` returns a negative value

4.2.9. getOffset

Prototype	`int getOffset() const;`
C API	`H5Tget_offset`
Description	Return the bit offset obtained from `H5Tget_offset` if it is not (-1).
Exception	`DataTypeIException` if `H5Tget_offset` returns (-1).

4.2.10. setOffset

Prototype	`void setOffset(size_t offset) const;`
C API	`H5Tset_offset`
Exception	`DataTypeIException` if `H5Tset_offset` returns a negative value

4.2.11. ~AtomType virtual destructor

Prototype	`virtual ~AtomType()`
Description	No operation

4.3. Attribute

4.3.1. Attribute copy constructor

Prototype	`Attribute(const Attribute& original): AbstractDs(original);`
Description	No operation

4.3.2. Attribute constructor using an existing id

Prototype	`Attribute(const hid_t attr_id): AbstractDs(attr_id)`
Description	Note: this function is used by CommonFG to return an `Attribute`. It should be considered to be a friend template function instead in the future.

4.3.3. read

Prototype	`void read(const DataType& mem_type, void *buf) const;`
	
	`void read(const DataType& mem_type, string& strg) const;`

C API	`H5Aread`
Description	These two functions are implemented separately. In the function that takes a `string`, a `char` pointer will be passed into `H5Aread` instead of a `void` buffer.
Exception	`AttributeIException` if `H5Aread` returns a negative value

4.3.4. write

Prototype	`void write(const DataType& mem_type, const void *buf) const;` `void write(const DataType& mem_type, const string& strg) const;`
C API	`H5Awrite`
Exception	`AttributeIException` if `H5Awrite` returns a negative value

4.3.5. getName

Prototype	`ssize_t getName(size_t buf_size, string& attr_name) const;`
	
	`string getName(size_t buf_size) const; // returns name,`

	not its length
C API	H5Aget_name
Description	The function getName, that returns a string, invokes the other getName. Missing getName that takes a char pointer.
Exception	AttributeIException if H5Aget_name returns a negative value

4.3.6. getSpace

Prototype	virtual DataSpace getSpace() const;
C API	H5Aget_space
Description	If H5Aget_space returns a positive value for dataspace id, create and return a DataSpace instance.
Exception	AttributeIException if H5Aget_space returns zero or a negative value

4.3.7. iterateAttrs

Prototype	int iterateAttrs()
Description	This function is a stub only, because DataSet inherits it from H5Object, but Attribute doesn't. It simply returns 0.

4.3.8. p_close

Prototype	virtual void p_close() const
Description	This function is removed due to the recent availability of the reference counting mechanism in the HDF5 library.

4.3.9. p_getType

Prototype	virtual hid_t p_getType() const
C API	H5Aget_type
Description	This function contains the common code that is used by getTypeClass and various datatype functions AbstractDs::getXxxType. It returns the positive data type id from H5Aget_type.
Exception	AttributeIException if H5Aget_type returns a zero or negative value

4.3.10. ~Attribute virtual destructor

Prototype	virtual ~Attribute()
Description	No operation

4.4. CommonFG

4.4.1. CommonFG default constructor

Prototype	CommonFG()
Description	No operation

4.4.2. createGroup

Prototype	Group createGroup(const string& name, size_t size_hint) const; Group createGroup(const char* name, size_t size_hint) const;
C API	H5Gcreate

Description	The `createGroup` that takes a `string` invokes the other `createGroup`.
Exception	`GroupIException` or `FileIException` , depending on whether this functions is invoked by `Group` of `H5File`, if `H5Gcreate` returns negative value

4.4.3. openGroup

Prototype	`Group openGroup(const string& name) const;`
	
	`Group openGroup(const char* name) const;`

C API	`H5Gopen`
Description	The `openGroup` that takes a `string` invokes the other `openGroup`.
Exception	`GroupIException` or `FileIException` , depending on whether this functions is invoked by `Group` of `H5File`, if `H5Gopen` returns negative value

4.4.4. createDataSet

Prototype	`DataSet createDataSet(const string& name, const DataType& data_type, const DataSpace& data_space, const DSetCreatPropList& create_plist = DSetCreatPropList::DEFAULT) const;`
	
	`DataSet createDataSet(const char* name, const DataType& data_type, const DataSpace& data_space, const DSetCreatPropList& create_plist = DSetCreatPropList::DEFAULT) const;`

C API	`H5Dcreate`
Description	The `createDataSet` that takes a `string` invokes the other `createDataSet`.
Exception	`GroupIException` or `FileIException` , depending on whether this functions is invoked by `Group` of `H5File`, if `H5Dcreate` returns negative value

4.4.5. openDataSet

Prototype	`DataSet openDataSet(const string& name) const;`
	
	`DataSet openDataSet(const char* name) const;`

C API	`H5Dopen`
Description	The `openDataSet` that takes a `string` invokes the other `openDataSet`.
Exception	`GroupIException` or `FileIException` , depending on whether this functions is invoked by `Group` of `H5File`, if `H5Dopen` returns negative value

4.4.6. openDataType

Prototype	`DataType openDataType(const string& name) const;`
	
	`DataType openDataType(const char* name) const;`

C API	`H5Topen`
Description	Invoke `p_openDataType` to obtain the datatype id, which is then used to

	construct a `DataType` instance. The `openDataType` that takes a `string` invokes the one that takes a `char` pointer.

4.4.7. openEnumType

Prototype	`EnumType openEnumType(const string& name) const;`
	
	`EnumType openEnumType(const char* name) const;`

C API	`H5Topen`
Description	Invoke `p_openDataType` to obtain the datatype id, which is then used to construct an `EnumType` instance. The `openDataType` that takes a `string` invokes the one that takes a `char` pointer.

4.4.8. openCompType

Prototype	`CompType openCompType(const string& name) const;`
	
	`CompType openCompType(const char* name) const;`

C API	`H5Topen`
Description	Invoke `p_openDataType` to obtain the datatype id, which is then used to construct a `CompType` instance. The `openDataType` that takes a `string` invokes the one that takes a `char` pointer.

4.4.9. openIntType

Prototype	`IntType openIntType(const string& name) const;`
	
	`IntType openIntType(const char* name) const;`

C API	`H5Topen`
Description	Invoke `p_openDataType` to obtain the datatype id, which is then used to construct an `IntType` instance. The `openDataType` that takes a `string` invokes the one that takes a `char` pointer.

4.4.10. openFloatType

Prototype	`FloatType openFloatType(const string& name) const;` `FloatType openFloatType(const char* name) const;`
C API	`H5Topen`
Description	Invoke `p_openDataType` to obtain the datatype id, which is then used to construct a `FloatType` instance. The `openDataType` that takes a `string` invokes the one that takes a `char` pointer.

4.4.11. openStrType

Prototype	`StrType openStrType(const string& name) const;`
	
	`StrType openStrType(const char* name) const;`

C API	`H5Topen`
Description	Invoke `p_openDataType` to obtain the datatype id, which is then used to construct a `StrType` instance. The `openDataType` that takes a `string` invokes the one that takes a `char` pointer.

4.4.12. link

Prototype	`void link(H5G_link_t link_type, const string& curr_name,`

	`const string& new_name) const;`
	
	`void link(H5G_link_t link_type, const char* curr_name, const char* new_name) const;`

C API	`H5Glink`
Description	The `link` that takes a `string` invokes the one that takes a `char` pointer.
Exception	`GroupIException` or `FileIException` , depending on whether this functions is invoked by `Group` of `H5File`, if `H5Glink` returns a negative value

4.4.13. unlink

Prototype	`void unlink(const string& name) const;`
	
	`void unlink(const char* name) const;`

C API	`H5Gunlink`
Description	The function `unlink` that takes a `string` invokes the one that takes a `char` pointer.
Exception	`GroupIException` or `FileIException` , depending on whether this functions is invoked by `Group` of `H5File`, if `H5Gunlink` returns a negative value

4.4.14. move

Prototype	`void move(const string& src, const string& dst) const;`
	
	`void move(const char* src, const char* dst) const;`

C API	`H5Gmove`
Description	The function `move` that takes a `string` invokes the one that takes a `char` pointer.
Exception	`GroupIException` or `FileIException` , depending on whether this functions is invoked by `Group` of `H5File`, if `H5Gmove` returns a negative value

4.4.15. getObjinfo

Prototype	`void getObjinfo(const string& name, hbool_t follow_link, H5G_stat_t& statbuf) const;`
	
	`void getObjinfo(const char* name, hbool_t follow_link, H5G_stat_t& statbuf) const;`

C API	`H5Gget_objinfo`
Description	The function `getObjinfo` that takes a `string` invokes the one that takes a `char` pointer.
Exception	`GroupIException` or `FileIException` , depending on whether this functions is invoked by `Group` of `H5File`, if `H5Gget_objinfo` returns a negative value

4.4.16. getLinkval

Prototype	`string getLinkval(const string& name, size_t size) const;`
	
	`string getLinkval(const char* name, size_t size) const;`

C API	H5Gget_linkval
Description	The function H5Gget_linkval that takes a string invokes the one that takes a char pointer.
Exception	GroupIException or FileIException , depending on whether this functions is invoked by Group of H5File, if H5Gget_linkval returns a negative value

4.4.17. setComment

Prototype	void setComment(const string& name, const string& comment) const;
	
	void setComment(const char* name, const char* comment) const;

C API	H5Gset_comment
Description	The function setComment that takes a string invokes the one that takes a char pointer.
Exception	GroupIException or FileIException , depending on whether this functions is invoked by Group of H5File, if H5Gset_comment returns negative value

4.4.18. getComment

Prototype	string getComment(const string& name, size_t bufsize) const;
	
	string getComment(const char* name, size_t bufsize) const;

C API	H5Gget_comment
Description	The function getComment that takes a string invokes the one that takes a char pointer.
Exception	GroupIException or FileIException , depending on whether this functions is invoked by Group of H5File, if H5Gget_comment returns negative value

4.4.19. mount

Prototype	void mount(const string& name, H5File& child, PropList& plist) const;
	
	void mount(const char* name, H5File& child, PropList& plist) const;

C API	H5Fmount
Description	The function mount that takes a string invokes the one that takes a char pointer.
Exception	GroupIException or FileIException , depending on whether this functions is invoked by Group of H5File, if H5Fmount returns negative value

4.4.20. unmount

Prototype	void unmount(const string& name) const;
	
	void unmount(const char* name) const;

C API	H5Funmount
Description	The function unmount function that takes a string invokes the one that takes a char pointer.
Exception	GroupIException or FileIException , depending on whether this functions is invoked by Group of H5File, if H5Funmount returns a negative value

4.4.21. iterateElems

Prototype	int iterateElems(const string& name, int *idx, H5G_iterate_t op, void *op_data);
	
	int iterateElems(const char* name, int *idx, H5G_iterate_t op, void *op_data);

C API	H5Giterate
Description	The function iterateElems that takes a string invokes the one that takes a char pointer.
Exception	GroupIException or FileIException , depending on whether this functions is invoked by Group of H5File, if H5Giterate returns a negative value

4.4.22. throwException

Prototype	throwException (const string& func_name, const string& msg) const = 0;
Description	pure virtual - implemented by H5File and Group so that each class can throw the appropriate exception when an error occurs within CommonFG.

4.4.23. ~CommonFG virtual destructor

Prototype	virtual ~CommonFG()
Description	No operation

4.5. CompType

4.5.1. CompType default constructor

Prototype	CompType() : DataType()
Description	No operation

4.5.2. CompType copy constructor

Prototype	CompType(const CompType& original) : DataType(original)
Description	No operation

4.5.3. CompType constructor using size

Prototype	CompType(size_t size) : DataType(H5T_COMPOUND, size)
Description	No operation

4.5.4. CompType constructor using an existing id

Prototype	CompType(const hid_t existing_id) : DataType(existing_id)
Description	No operation.

4.5.5. CompType from a DataSet

Prototype	`CompType(const DataSet& dataset) : DataType()`
C API	`H5Dget_type`
Exception	`DataTypeIException` if `H5Dget_type` returns a zero or a negative value.

4.5.6. getNmembers

Prototype	`int getNmembers() const;`
C API	`H5Tget_nmembers`
Description	Returns the number of members obtained from `H5Tget_nmembers`.
Exception	`DataTypeIException` if `H5Tget_nmembers` returns a negative value

4.5.7. getMemberName

Prototype	`string getMemberName(int member_num) const;`
C API	`H5Tget_member_name`
Description	If H5Tget_member_name returns a non-NULL member name, then return it as a `string`.
Exception	`DataTypeIException` if `H5Tget_member_name` returns a NULL char pointer .

4.5.8. getMemberIndex

Prototype	`int getMemberIndex(const char* name) const;` `int getMemberIndex(const string& name) const;`
C API	`H5Tget_member_index`
Description	Returns the index of member obtained from `H5Tget_member_index`.The function `getMemberIndex` that takes a `string` for name invokes the one that takes a `char` pointer.
Exception	`DataTypeIException` if `H5Tget_member_index` returns a negative value

4.5.9. getMemberOffset

Prototype	`size_t getMemberOffset(int memb_no) const;`
C API	`H5Tget_member_offset`
Description	Simply return the offset value obtained from `H5Tget_member_offset`. Note: 0 is not a failure.

4.5.10. getMemberDims

Prototype	`int getMemberDims(int member_num, size_t* dims, int* perm) const;`
Description	The C API for this function is no longer supported. This function will be removed after announcing in the release notes.

4.5.11. getMemberClass

Prototype	`H5T_class_t getMemberClass(int member_num) const;`
C API	`H5Tget_class and H5Tget_member_type`
Description	First, get the datatype of the member with H5Tget_member_type, then get the type class of the member with `H5Tget_class`.
Exception	`DataTypeIException` if `H5Tget_member_type` returns a zero or a negative value, or if `H5Tget_class` returns `H5T_NO_CLASS`.

4.5.12. p_getMemberDataType

Prototype	`hid_t p_get_member_type(int member_num) const`
C API	`H5Tget_member_type`
Description	This function is private. It returns the member's type identifier obtained from `H5Tget_member_type`, when the identifier is a positive value.

4.5.13. getMemberDataType

Prototype	`DataType getMemberDataType(int member_num) const;`
Description	Use the private member function `p_get_member_type` to get the identifier of the specified member, then construct and return a `DataType` instance using that identifier.

4.5.14. getMemberEnumType

Prototype	`EnumType getMemberEnumType(int member_num) const;`
Description	Use the private member function `p_get_member_type` to get the identifier of the specified member, then construct and return an `EnumType` instance using that identifier.

4.5.15. getMemberCompType

Prototype	`CompType getMemberCompType(int member_num) const;`
Description	Use the private member function `p_get_member_type` to get the identifier of the specified member, then construct and return an `CompType` instance using that identifier.

4.5.16. getMemberIntType

Prototype	`IntType getMemberIntType(int member_num) const;`
Description	Use the private member function `p_get_member_type` to get the identifier of the specified member, then construct and return an `IntType` instance using that identifier.

4.5.17. getMemberFloatType

Prototype	`FloatType getMemberFloatType(int member_num) const;`
Description	Use the private member function `p_get_member_type` to get the identifier of the specified member, then construct and return an `FloatType` instance using that identifier.

4.5.18. getMemberStrType

Prototype	`StrType getMemberStrType(int member_num) const;`
Description	Use the private member function `p_get_member_type` to get the identifier of the specified member, then construct and return an `StrType` instance using that identifier.

4.5.19. insertMember

Prototype	`void insertMember(const string& name, size_t offset, const DataType& new_member) const;`
C API	H5Tinsert

Description	The ability to insert an array is removed from this member function(???)
Exception	`DataTypeIException` if `H5Tinsert` returns a negative value

4.5.20. pack

Prototype	`void pack() const;`
C API	H5Tpack
Exception	`DataTypeIException` if `H5Tpack` returns a negative value

4.5.21. ~CompType virtual destructor

Prototype	`virtual ~H5File()`
Description	No operation

4.6. DataSet

4.6.1. DataSet default constructor

Prototype	`DataSet() : AbstractDs()`
Description	No operation

4.6.2. DataSet copy constructor

Prototype	`DataSet(const DataSet& original) : AbstractDs(original)`
Description	No operation

4.6.3. DataSet constructor using an existing id

Prototype	`DataSet(const hid_t dataset_id) : AbstractDs(dataset_id)`
Description	No operation. Note: this constructor is used by CommonFG to return a DataSet instance. It should be considered to be a friend template function instead in the future.

4.6.4. getSpace

Prototype	`virtual DataSpace getSpace() const;`
C API	H5Dget_space
Description	This function creates and returns a `DataSpace` instance, using the dataspace id returned by `H5Dget_space`.
Exception	`DataSetIException` if H5Dget_space returns a negative value

4.6.5. getCreatePlist

Prototype	`DSetCreatPropList getCreatePlist() const;`
C API	`H5Dget_create_plist`
Description	This function constructs and returns a `DSetCreatPropList` instance, using the dataspace id returned by `H5Dget_create_plist`.
Exception	`DataSetIException` if `H5Dget_create_plist` returns a negative value

4.6.6. getStorageSize

Prototype	`hsize_t getStorageSize() const;`
C API	`H5Dget_storage_size`
Description	Returns a positive value for storage size from `H5Dget_storage_size`
Exception	`DataSetIException` if `H5Dget_storage_size` returns a zero or

	negative value

4.6.7. getVlenBufSize

Prototype	`hsize_t getVlenBufSize(DataType& type, DataSpace& space) const;`
Description	Simply throws an exception indicating that the function is not yet implemented.
Exception	`DataSetIException`

4.6.8. vlenReclaim

Prototype	`void vlenReclaim(DataType& type, DataSpace& space, DSetMemXferPropList& xfer_plist, void* buf) const`
C API	`H5Dvlen_reclaim`
Exception	`DataSetIException` if `H5Dvlen_reclaim` returns a negative value

4.6.9. read

Prototype	`void read(void* buf, const DataType& mem_type, const DataSpace& mem_space = DataSpace::ALL, const DataSpace& file_space = DataSpace::ALL, const DSetMemXferPropList& xfer_plist = DSetMemXferPropList::DEFAULT) const;`
	
	`void read(string& buf, const DataType& mem_type, const DataSpace& mem_space = DataSpace::ALL, const DataSpace& file_space = DataSpace::ALL, const DSetMemXferPropList& xfer_plist = DSetMemXferPropList::DEFAULT) const;`

C API	`H5Dread`
Description	The `read` function that takes a `string` invokes the `read` function that takes a `void` pointer.
Exception	`DataSetIException` if `H5Dread` returns a negative value

4.6.10. write

Prototype	`void write(const void* buf, const DataType& mem_type, const DataSpace& mem_space = DataSpace::ALL, const DataSpace& file_space = DataSpace::ALL, const DSetMemXferPropList& xfer_plist = DSetMemXferPropList::DEFAULT) const;`
	
	`void write(const string& buf, const DataType& mem_type, const DataSpace& mem_space = DataSpace::ALL, const DataSpace& file_space = DataSpace::ALL, const DSetMemXferPropList& xfer_plist = DSetMemXferPropList::DEFAULT) const;`

C API	`H5Dwrite`
Description	The `write` function that takes a `string` invokes the `write` function that takes a `void` pointer.
Exception	`DataSetIException` if `H5Dwrite` returns a negative value

4.6.11. iterateElems

Prototype	`int iterateElems(void* buf, const DataType& type, const DataSpace& space, H5D_operator_t op, void* op_data = NULL);`

C API	H5Diterate
Description	This function is currently not implemented in C++ style yet. In addition, it may be moved to class DataSpace since it is iterating over elements that are in a dataspace.
Exception	DataSetIException if H5Diterate returns a negative value

4.6.12. extend

Prototype	void extend (const hsize_t* size) const;
C API	H5Dextend
Exception	DataSetIException if H5Dextend returns a negative value

4.6.13. fillMemBuf

Prototype	void fillMemBuf(void *buf, DataType& buf_type, DataSpace& space);
	
	void fillMemBuf(const void *fill, DataType& fill_type, void *buf, DataType& buf_type, DataSpace& space);

C API	H5Dfill
Description	The fillMemBuf that takes a void pointer as its first argument passes arguments to H5Dfill as followed: H5Dfill(NULL, buf_type_id, buf, buf_type_id, space_id); The fillMemBuf that takes a const void pointer as its first argument passes arguments to H5Dfill as followed: H5Dfill(fill, fill_type_id, buf, buf_type_id, space_id);
Exception	DataSetIException if H5Dfill returns a negative value

4.6.14. p_getType

Prototype	virtual hid_t p_getType() const
C API	H5Dget_type
Description	This function contains the common code that is used by getTypeClass and various datatype functions AbstractDs::getXxxType. It returns the positive data type id from H5Dget_type.
Exception	DataSetIException if H5Dget_type returns a zero or negative value

4.6.15. p_close

Prototype	virtual void p_close() const
Description	This function is removed due to the recent availability of the reference counting mechanism in the HDF5 library.

4.6.16. ~DataSet virtual destructor

Prototype	virtual ~H5File()
Description	No operation

Notes on implementation:

1. read and write may be implemented using operators >> and << in the future version of the C++ API.

2. iterateElems is not yet implemented.

4.7. DataSpace

4.7.1. DataSpace default constructor

Prototype	`DataSpace() : IdComponent()`
Description	No operation

4.7.2. DataSpace copy constructor

Prototype	`DataSpace(const DataSpace& original) : IdComponent(original);`

4.7.3. DataSpace constructor using a type

Prototype	`DataSpace(H5S_class_t type);`
C API	`H5Screate`
Exception	`DataSpaceIException` if `H5Screate` returns a zero or a negative value

4.7.4. DataSpace constructor using dimension information

Prototype	`DataSpace(int rank, const hsize_t * dims, const hsize_t * maxdims = NULL);`
C API	`H5Screate_simple`
Exception	`DataSpaceIException` if `H5Screate_simple` returns a zero or a negative value

4.7.5. DataSpace constructor using an existing id

Prototype	`DataSpace(const hid_t space_id) : IdComponent(space_id);`
Description	No operation

4.7.6. copy

Prototype	`void copy(const DataSpace& like_space);`
C API	`H5Scopy`
Description	Reset the identifier of this instance before copying `like_space` to it, to make certain that the previous dataspace is properly closed.
Exception	`DataSpaceIException` if `H5Scopy` returns a zero or a negative value

4.7.7. operator=

Prototype	`DataSpace& operator=(const DataSpace& rhs);`
Description	Invoke member function `copy` to copy `rhs` to `this`, then return *`this`.

4.7.8. isSimple

Prototype	`bool isSimple() const;`
C API	`H5Sis_simple`
Description	If `H5Sis_simple` returns • a positive value, then return true • a zero, then return false, • others, then throw exception
Exception	`DataSpaceIException` if `H5Sis_simple` returns a negative value

4.7.9. offsetSimple

Prototype	`void offsetSimple(const hssize_t* offset) const;`
C API	`H5Soffset_simple`
Exception	`DataSpaceIException` if `H5Soffset_simple` returns a negative value

4.7.10. getSimpleExtentDims

Prototype	`int getSimpleExtentDims(hsize_t *dims, hsize_t *maxdims = NULL) const;`
C API	`H5Sget_simple_extent_dims`
Description	Return the dimension size from `H5Sget_simple_extent_dims` if it is not a negative value.
Exception	`DataSpaceIException` if `H5Sget_simple_enxtent_dims` returns a negative value

4.7.11. getSimpleExtentNdims

Prototype	`int getSimpleExtentNdims() const;`
C API	`H5Sget_simple_extent_ndims`
Description	Return the dimensionality dimension size from `H5Sget_simple_extent_ndims` if it is a non-negative value.
Exception	`DataSpaceIException` if `H5Sget_simple_enxtent_ndims` returns a negative value

4.7.12. getSimpleExtentNpoints

Prototype	`hssize_t getSimpleExtentNpoints() const;`
C API	`H5Sget_simple_extent_npoints`
Description	Return the number of elements obtained from the function H5Sget_simple_extent_npoints, if it is a non-negative value.
Exception	`DataSpaceIException` if `H5Sget_simple_enxtent_npoints` returns negative value

4.7.13. getSimpleExtentType

Prototype	`H5S_class_t getSimpleExtentType() const;`
C API	`H5Sget_simple_extent_type`
Description	Return the class obtained from the function `H5Sget_simple_extent_type`, if it is not equal to `H5S_NO_CLASS`.
Exception	`DataSpaceIException` if `H5Sget_simple_enxtent_type` returns `H5S_NO_CLASS`.

4.7.14. extentCopy

Prototype	`void extentCopy(DataSpace& dest_space) const;`
C API	`H5Sextent_copy`
Exception	`DataSpaceIException` if `H5Sextent_copy` returns a negative value

4.7.15. setExtentSimple

Prototype	`void setExtentSimple(int rank, const hsize_t *current_size, const hsize_t *maximum_size = NULL) const;`
C API	`H5Sset_extent_simple`
Exception	`DataSpaceIException` if `H5Sset_extent_simple` returns a negative value

4.7.16. setExtentNone

Prototype	`void setExtentNone() const;`
C API	`H5Sset_extent_none`
Exception	`DataSpaceIException` if `H5Sset_extent_none` returns a negative value

4.7.17. getSelectNpoints

Prototype	`hssize_t getSelectNpoints() const;`
C API	`H5Sget_select_npoints`
Description	Return the number of elements obtained from the function H5Sget_select_npoints, if it is a non-negative value.
Exception	`DataSpaceIException` if `H5Sget_select_npoints` returns a negative value

4.7.18. getSelectHyperNblocks

Prototype	`hssize_t getSelectHyperNblocks() const;`
C API	`H5Sget_select_hyper_nblocks`
Description	Return the number of hyperslab blocks obtained from the function H5Sget_select_hyper_nblocks, if it is a non-negative value.
Exception	`DataSpaceIException` if `H5Sget_select_hyper_nblocks` returns a negative value

4.7.19. getSelectHyperBlocklist

Prototype	`void getSelectHyperBlocklist(hsize_t startblock, hsize_t numblocks, hsize_t *buf) const;`
C API	`H5Sget_select_hyper_blocklist`
Exception	`DataSpaceIException` if `H5Sget_select_hyper_blocklist` returns a negative value

4.7.20. getSelectElemNpoints

Prototype	`hssize_t getSelectElemNpoints() const;`
C API	`H5Sget_select_elem_npoints`
Description	Return the number of element points obtained from the function H5Sget_select_elem_npoints, if it is a non-negative value.
Exception	`DataSpaceIException` if H5Sget_select_elem_npoints returns a negative value

4.7.21. getSelectElemPointlist

Prototype	`void getSelectElemPointlist(hsize_t startpoint, hsize_t numpoints, hsize_t *buf) const;`
C API	`H5Sget_select_elem_pointlist`
Exception	`DataSpaceIException` if H5Sget_select_elem_pointlist returns a negative value

4.7.22. getSelectBounds

Prototype	`void getSelectBounds(hssize_t* start, hssize_t* end) const;`

C API	H5Sget_select_bounds
Exception	DataSpaceIException if H5Sget_select_bounds returns a negative value

4.7.23. selectElements

Prototype	void selectElements(H5S_seloper_t op, const size_t num_elements, const hssize_t *coord[ ]) const;
C API	H5Sselect_elements
Exception	DataSpaceIException if H5Sselect_elements returns a negative value

4.7.24. selectAll

Prototype	void selectAll() const;
C API	H5Sselect_all
Exception	DataSpaceIException if H5Sselect_all returns a negative value

4.7.25. selectNone

Prototype	void selectNone() const;
C API	H5Sselect_none
Exception	DataSpaceIException if H5Sselect_none returns a negative value

4.7.26. selectValid

Prototype	bool selectValid() const;
C API	H5Sselect_valid
Description	If H5Sselect_valid returns - a positive value, then return true, - a zero, then return false, - others, then raise an exception
Exception	DataSpaceIException if H5Sselect_valid returns a negative value

4.7.27. selectHyperslab

Prototype	void selectHyperslab(H5S_seloper_t op, const hsize_t *count, const hssize_t *start, const hsize_t *stride = NULL, const hsize_t *block = NULL) const;
C API	H5Sselect_hyperslab
Exception	DataSpaceIException if H5Sselect_hyperslab returns a negative value

4.7.28. p_close

Prototype	virtual void p_close() const
Description	This function is removed due to the recent availability of the reference counting mechanism in the HDF5 library.

4.7.29. ~DataSpace virtual destructor

Prototype	virtual ~DataSpace()
Description	No operation

4.8. DataType

4.8.1. DataType default constructor

Prototype	`DataType(): H5Object(), is_predtype(false)`
Description	No operation

4.8.2. DataType copy constructor

Prototype	`DataType(const DataType& original);`
Description	Copy the data member `is_predtype` from `original`.

4.8.3. DataType constructor using an existing id

Prototype	`DataType(const hid_t existing_id, bool predtype = false) : H5Object(existing_id), is_predtype(predtype)`
Description	No operation

4.8.4. DataType constructor using a type class

Prototype	`DataType(const H5T_class_t type_class, size_t size);`
C API	`H5Tcreate`
Exception	DataTypeIException if `H5Tcreate` returns a negative value

4.8.5. operator=

Prototype	`DataType& operator=(const DataType& rhs);`
Description	Invoke member function `copy` to copy `rhs` to `this`, then return `*this`.

4.8.6. operator==

Prototype	`bool operator==(const DataType& compared_type) const;`
C API	`H5Tequal`
Description	If H5Tequal returns • a positive value, then return true, • a zero, then return false, • others, then raise an exception
Exception	DataTypeIException if `H5Tcreate` returns a negative value

4.8.7. copy

Prototype	`void copy(const DataType& like_type);`
C API	`H5Tcopy`
Description	Reset the identifier of this instance before copying `like_type` to it, to make certain that the previous datatype is properly closed.
Exception	`DataTypeIException` if `H5Tcopy` returns a zero or a negative value.

4.8.8. getClass

Prototype	`H5T_class_t getClass() const;`
C API	`H5Tget_class`
Exception	`DataTypeIException` if `H5Tget_class` returns `H5T_NO_CLASS`.

4.8.9. commit

Prototype	`void commit(H5Object& loc, const string& name) const;`
	
	`void commit(H5Object& loc, const char* name) const;`

C API	`H5Tcommit`
Description	The function `commit` that takes a `string` invokes the one that takes a

	`char` pointer.
Exception	`DataTypeIException` if `H5Tcommit` returns a negative value

4.8.10. committed

Prototype	`bool committed() const;`
C API	`H5Tcommitted`
Description	If H5Tcommitted returns • a positive value, then return true, • a zero, then return false, • others, then raise an exception
Exception	`DataTypeIException` if `H5Tcommitted` returns a negative value

4.8.11. find

Prototype	`H5T_conv_t find(const DataType& dest, H5T_cdata_t **pcdata) const;`
C API	`H5Tfind`
Exception	`DataTypeIException` if `H5Tfind` returns a negative value

4.8.12. convert

Prototype	`void convert(const DataType& dest, hsize_t nelmts, void *buf, void *background, PropList& plist) const;`
C API	`H5Tconvert`
Exception	`DataTypeIException` if `H5Tconvert` returns a negative value

4.8.13. setOverflow

Prototype	`void setOverflow(H5T_overflow_t func) const;`
C API	`H5Tset_overflow`
Exception	`DataTypeIException` if `H5Tset_overflow` returns a negative value.

4.8.14. getOverflow

Prototype	`H5T_overflow_t getOverflow(void) const;`
C API	`H5Tget_overflow`
Description	Simply return what `H5Tget_overflow` returns.

4.8.15. lock

Prototype	`void lock() const;`
C API	`H5Tlock`
Exception	`DataTypeIException` if `H5Tlock` returns a negative value.

4.8.16. getSize

Prototype	`size_t getSize() const;`
C API	`H5Tget_size`
Exception	`DataTypeIException` if `H5Tget_size` returns a zero or a negative value

4.8.17. getSuper

Prototype	`DataType getSuper() const;`
C API	`H5Tget_super`

Description	If `H5Tget_super` returns a valid datatype id, create and return the base type, otherwise, raise exception. Note: not quite right for specific types yet.
Exception	`DataTypeIException` if `H5Tget_super` returns a zero or a negative value

4.8.18. registerFunc

Prototype	`void registerFunc(H5T_pers_t pers, const string& name, const DataType& dest, H5T_conv_t func) const;`
	
	`void registerFunc(H5T_pers_t pers, const char* name, const DataType& dest, H5T_conv_t func) const;`

C API	`H5Tregister`
Description	The function `registerFunc` that takes a `string` for `name` invokes the function that takes a `char` pointer.
Exception	`DataTypeIException` if `H5Tregister` returns a negative value.

4.8.19. unregister

Prototype	`void unregister(H5T_pers_t pers, const string& name, const DataType& dest, H5T_conv_t func) const;`
	
	`void unregister(H5T_pers_t pers, const char* name, const DataType& dest, H5T_conv_t func) const;`

C API	`H5Tunregister`
Description	The function `registerFunc` that takes a `string` for `name` invokes the function that takes a `char` pointer.
Exception	`DataTypeIException` if `H5Tunregister` returns a negative value

4.8.20. setTag

Prototype	`void setTag(const string& tag) const;`
	
	`void setTag(const char* tag) const;`

C API	`H5Tset_tag`
Description	The function `setTag` that takes a `string` for `tag` invokes the function that takes a `char` pointer.
Exception	`DataTypeIException` if `H5Tset_tag` returns a negative value

4.8.21. getTag

Prototype	`string getTag() const;`
C API	`H5Tget_tag`
Description	If `H5Tget_tag` returns a non-NULL char pointer, then convert the char pointer to string then return it.
Exception	`DataTypeIException` if `H5Tget_tag` returns a NULL pointer.

4.8.22. ~DataType virtual destructor

Prototype	`virtual ~DataType()`
Description	No operation

4.8.23. p_close

Prototype	`void p_close() const`
Description	This function is removed due to the recent availability of the reference counting mechanism in the HDF5 library.

4.9. DSetCreatPropList

4.9.1. DSetCreatPropList default constructor

Prototype	`DSetCreatPropList() : PropList(H5P_DATASET_CREATE)`
Description	No operation

4.9.2. DSetCreatPropList copy constructor

Prototype	`DSetCreatPropList(const DSetCreatPropList& orig) : PropList(orig)`
Description	No operation

4.9.3. DSetCreatPropList constructor using an existing id

Prototype	`DSetCreatPropList(const hid_t plist_id) : PropList(plist_id)`
Description	No operation

4.9.4. setChunk

Prototype	`void setChunk(int ndims, const hsize_t* dim) const;`
C API	`H5Pset_chunk`
Exception	`PropListIException` if H5Pset_chunk returns a negative value

4.9.5. getChunk

Prototype	`int getChunk(int max_ndims, hsize_t* dim) const;`
C API	`H5Pget_chunk`
Description	Return the chunk size obtained from `H5Pget_chunk` if it is a non-negative value.
Exception	`PropListIException` if H5Pget_chunk returns a negative value

4.9.6. setLayout

Prototype	`void setLayout(hid_t plist, H5D_layout_t layout) const;`
C API	`H5Pset_layout`
Exception	`PropListIException` if H5Pset_layout returns a negative value

4.9.7. getLayout

Prototype	`H5D_layout_t getLayout() const;`
C API	`H5Pget_layout`
Description	Return the layout obtained from `H5Pget_layout` if it is a non-negative value.
Exception	`PropListIException` if H5Pget_layout returns a negative value

4.9.8. setDeflate

Prototype	`void setDeflate(int level) const;`
C API	`H5Pset_deflate`
Exception	`PropListIException` if H5Pset_deflate returns a negative value

4.9.9. setFillValue

Prototype	`void setFillValue(DataType& fvalue_type, const void* value) const;`
C API	`H5Pset_fill_value`
Exception	`PropListIException` if H5Pset_fill_value returns a negative value

4.9.10. getFillValue

Prototype	`void getFillValue(DataType& fvalue_type, void* value) const;`
C API	`H5Pget_fill_value`
Exception	`PropListIException` if H5Pget_fill_value returns a negative value

4.9.11. setFilter

Prototype	`void setFilter(H5Z_filter_t filter, unsigned int flags, size_t cd_nelmts, const unsigned int cd_values[]) const;`
C API	`H5Pset_filter`
Exception	`PropListIException` if H5Pset_filter returns a negative value

4.9.12. getNfilters

Prototype	`int getNfilters() const;`
C API	`H5Pget_nfilters`
Description	Return the number of filters obtained from `H5Pget_nfilters` if it is a non-negative value.
Exception	`PropListIException` if H5Pget_nfilter returns negative value

4.9.13. getFilter

Prototype	`H5Z_filter_t getFilter(int filter_number, unsigned int& flags, size_t& cd_nelmts, unsigned int* cd_values, size_t namelen, char name[]) const;`
C API	`H5Pget_filter`
Description	Return the number of filters obtained from `H5Pget_filters` if it is not equal to `H5Z_FILTER_ERROR`.
Exception	`PropListIException` if H5Pget_filter returns `H5Z_FILTER_ERROR`

4.9.14. setExternal

Prototype	`void setExternal(const char* name, off_t offset, hsize_t size) const;`
C API	`H5Pset_external`
Exception	`PropListIException` if H5Pset_external returns a negative value

4.9.15. getExternalCount

Prototype	`int getExternalCount() const;`
C API	`H5Pget_external_count`
Description	Return the number of external files obtained from `H5Pget_external_count` if it is a non-negative value.
Exception	`PropListIException` if H5Pset_external_count returns a negative value

4.9.16. getExternal

Prototype	`void getExternal(int idx, size_t name_size, char* name, off_t& offset, hsize_t& size) const;`
C API	`H5Pget_external`
Exception	`PropListIException` if H5Pget_external returns a negative value

4.9.17. ~DSetCreatPropList virtual destructor

Prototype	`virtual ~DSetCreatPropList()`
Description	No operation

4.10. DSetMemXferPropList

4.10.1. DSetMemXferPropList default constructor

Prototype	`DSetMemXferPropList() : PropList(H5P_DATASET_XFER)`
Description	No operation

4.10.2. DSetMemXferPropList copy constructor

Prototype	`DSetMemXferPropList(const DSetMemXferPropList& orig) : PropList(orig);`
Description	No operation

4.10.3. DSetMemXferPropList constructor using an existing id

Prototype	`DSetMemXferPropList (const hid_t plist_id) : PropList(plist_id)`
Description	No operation

4.10.4. setBuffer

Prototype	`void setBuffer(size_t size, void* tconv, void* bkg) const;`
C API	`H5Pset_buffer`
Exception	`PropListIException` if `H5Pset_buffer` returns a negative value

4.10.5. getBuffer

Prototype	`size_t getBuffer(void** tconv, void** bkg) const;`
C API	`H5Pget_buffer`
Description	Return the buffer settings obtained from `H5Pget_buffer` if it is not equal to 0.
Exception	`PropListIException` if `H5Pget_buffer` returns 0.

4.10.6. setPreserve

Prototype	`void setPreserve(bool status) const;`
C API	`H5Pset_preserve`
Exception	`PropListIException` if H5Pset_preserve returns a negative value

4.10.7. getPreserve

Prototype	`bool getPreserve() const;`
C API	`H5Pget_preserve`
Description	If `H5Pget_preserve` returns • a positive value, then return true,

	• a zero, then return false, • others, raise an exception
Exception	`PropListIException` if H5Pget_preserve returns a negative value

4.10.8. setBtreeRatios

Prototype	`void setBtreeRatios(double left, double middle, double right) const;`
C API	`H5Pset_btree_ratios`
Exception	`PropListIException` if H5Pset_btree_ratios returns a negative value

4.10.9. getBtreeRatios

Prototype	`void getBtreeRatios(double& left, double& middle, double& right) const;`
C API	`H5Pget_btree_ratios`
Exception	`PropListIException` if H5Pget_btree_ratios returns a negative value

4.10.10. setVlenMemManager

Prototype	`void setVlenMemManager() const;` `void setVlenMemManager(H5MM_allocate_t alloc, void* alloc_info, H5MM_free_t free, void* free_info) const;`
C API	`H5Pset_vlen_mem_manager`
Description	The function `setVlenMemManager` that takes no parameters invokes the other `setVlenMemManager` with all parameters set to `NULL`.
Exception	`PropListIException` if H5Pset_vlen_mem_manager returns a negative value

4.10.11. ~DSetMemXferPropList virtual destructor

Prototype	`virtual ~DSetMemXferPropList()`
Description	No operation

4.11. EnumType

4.11.1. EnumType default constructor

Prototype	`EnumType(): DataType() {}`
Description	No operation

4.11.2. EnumType copy constructor

Prototype	`EnumType(const EnumType& original) : DataType(original)`
Description	No operation

4.11.3. EnumType constructor using size

Prototype	`EnumType(size_t size) : DataType(H5T_ENUM, size)`
Description	No operation

4.11.4. EnumType constructor from a dataset

Prototype	`EnumType(const DataSet& dataset);`
C API	`H5Dget_type`
Exception	`DataTypeIException` if `H5Dget_type` returns a zero or a negative value.

4.11.5. EnumType constructor from an IntType

Prototype	`EnumType(const IntType& data_type);`
C API	`H5Tenum_create`
Exception	`DataTypeIException` if `H5Tenum_create` returns a zero or a negative value.

4.11.6. EnumType constructor using an existing id

Prototype	`EnumType(const hid_t existing_id) : DataType(existing_id)`
Description	No operation

4.11.7. insert

Prototype	`void insert(const string& name, void *value) const;`
	
	`void insert(const char* name, void *value) const;`

C API	`H5Tenum_insert`
Description	The function `insert` that takes a `string` for name invokes the one that takes a `char` pointer.
Exception	`DataTypeIException` if `H5Tenum_insert` returns a negative value

4.11.8. nameOf

Prototype	`string nameOf(void *value, size_t size) const;`
C API	`H5Tenum_nameof`
Description	Returns the name from `H5Tenum_nameof` as a `string`.
Exception	`DataTypeIException` if `H5Tenum_nameof` returns a negative value

4.11.9. valueOf

Prototype	`void valueOf(const string& name, void *value) const;`
	
	`void valueOf(const char* name, void *value) const;`

C API	`H5Tenum_valueof`
Description	The function `H5Tenum_valueof` that takes a `string` for name invokes the one that takes a `char` pointer.
Exception	`DataTypeIException` if `H5Tenum_valueof` returns a negative value

4.11.10. getMemberIndex

Prototype	`int getMemberIndex(const char* name) const;`
	
	`int getMemberIndex(const string& name) const;`

C API	`H5Tget_member_index`
Description	Returns the index of member obtained from `H5Tget_member_index`. The function getMemberIndex that takes a string for name invokes the one that takes a char pointer.
Exception	`DataTypeIException` if `H5Tget_member_index` returns a negative value

4.11.11. getMemberValue

Prototype	`void getMemberValue(int memb_no, void *value) const;`

C API	H5Tget_member_value
Exception	DataTypeIException if H5Tget_member_value returns a negative value.

4.11.12. ~EnumType virtual destructor

Prototype	virtual ~H5File()
Description	No operation

4.12. Exception

4.12.1. Exception default constructor

Prototype	Exception() : detail_message(""), func_name("")
Description	No operation

4.12.2. Exception copy constructor

Prototype	Exception(const Exception& orig)
Description	It copies both members detail_message and func_name, from the orig object.

4.12.3. Exception constructor with data members

Prototype	Exception(const string& func_name, const string& message) Exception::Exception(const char* func_name, const char* message)
Description	Initialize the members detail_message and func_name, with the given parameters.

4.12.4. getMajorString

Prototype	string getMajorString(hid_t err_major_id) const;
C API	H5Eget_msg
Description	It passes a major error number to the C API to obtain the character string that describes the major error.

4.12.5. getMinorString

Prototype	string getMinorString(hid_t err_minor_id) const;
C API	H5Eget_msg
Description	It passes a minor error number to the C API to obtain the character string that describes the minor error.

4.12.6. getFuncName

Prototype	string getFuncName() const;
	
	const char* getCFuncName() const;

Description	Returns the member func_name.

4.12.7. getDetailMsg

Prototype	string getDetailMsg() const;
Description	Returns the value of the member detail_message as a string.

4.12.8. getCDetailMsg

Prototype	`const char* getCFuncName() const;`
Description	Returns the value of the member `detail_message` as a character pointer.

4.12.9. setAutoPrint

Prototype	`static void setAutoPrint(H5E_auto_t func, void* client_data);`
C API	`H5Eset_auto`
Exception	`Exception` if `H5Eset_auto` returns a negative value

4.12.10. dontPrint

Prototype	`static void dontPrint();`
C API	`H5Eset_auto`
Description	It passes NULL for both arguments into `H5Eset_auto`
Exception	`Exception` if `H5Eset_auto` returns a negative value

4.12.11. getAutoPrint

Prototype	`static void getAutoPrint(H5E_auto_t& func, void** client_data);`
C API	`H5Eget_auto`
Exception	`Exception` if `H5Eget_auto` returns a negative value

4.12.12. clearErrorStack

Prototype	`static void clearErrorStack();`
C API	`H5Eclear`
Exception	`Exception` if `H5Eclear` returns a negative value

4.12.13. walkErrorStack

Prototype	`static void walkErrorStack(H5E_direction_t direction, H5E_walk_t func, void* client_data);`
C API	`H5Ewalk`
Exception	`Exception` if `H5Ewalk` returns a negative value

4.12.14. printError

Prototype	`virtual void printError(FILE* stream = NULL) const;`
C API	`H5Eprint`
Exception	`Exception` if `H5Ewalk` returns negative value

4.12.15.

4.12.16. ~Exception virtual destructor

Prototype	`virtual ~Exception()`
Description	No operation

4.13. FileAccPropList

4.13.1. FileAccPropList default constructor

Prototype	`FileAccPropList(): PropList(H5P_FILE_ACCESS)`
Description	No operation

4.13.2. FileAccPropList copy constructor

Prototype	`FileAccPropList(const FileAccPropList& orig) : PropList(orig);`
Description	No operation

4.13.3. FileAccPropList constructor using an existing id

Prototype	`FileAccPropList (const hid_t plist_id) : PropList(plist_id)`
Description	No operation

4.13.4. setCache

Prototype	`void setCache(int mdc_nelmts, size_t rdcc_nelmts, size_t rdcc_nbytes, double rdcc_w0) const;`
C API	`H5Pset_cache`
Exception	`PropListIException` if `H5Pset_cache` returns a negative value

4.13.5. getCache

Prototype	`void getCache(int& mdc_nelmts, size_t& rdcc_nelmts, size_t& rdcc_nbytes, double& rdcc_w0) const;`
C API	`H5Pget_cache`
Exception	`PropListIException` if `H5Pget_cache` returns a negative value

4.13.6. setAlignment

Prototype	`void setAlignment(hsize_t threshold = 1, hsize_t alignment = 1) const;`
C API	`H5Pset_alignment`
Exception	`PropListIException` if H5Pset_alignment returns a negative value

4.13.7. getAlignment

Prototype	`void getAlignment(hsize_t& threshold, hsize_t& alignment) const;`
C API	`H5Pget_alignment`
Exception	`PropListIException` if H5Pget_alignment returns a negative value

4.13.8. setGcReferences

Prototype	`void setGcReferences(unsigned gc_ref = 0) const;`
C API	`H5Pset_gc_references`
Exception	`PropListIException` if H5Pset_gc_references returns a negative value

4.13.9. getGcReferences

Prototype	`unsigned getGcReferences() const;`
C API	`H5Pget_gc_references`
Description	Return garbage collecting references setting obtained from `H5Pget_gc_references`, if it is a non-negative value.
Exception	`PropListIException` if H5Pget_gc_references returns negative value

4.13.10. ~FileAccPropList virtual destructor

Prototype	`virtual ~FileAccPropList()`
Description	No operation

4.14. FileCreatPropList

4.14.1. FileCreatPropList default constructor

Prototype	`FileCreatPropList(): PropList(H5P_FILE_CREATE)`
Description	No operation

4.14.2. FileCreatPropList copy constructor

Prototype	`FileCreatPropList(const FileCreatPropList& original) : PropList(original)`
Description	No operation

4.14.3. FileCreatPropList constructor using an existing id

Prototype	`FileCreatPropList (const hid_t plist_id) : PropList(plist_id)`
Description	No operation

4.14.4. getVersion

Prototype	`void getVersion(unsigned& boot, unsigned& freelist, unsigned& stab, unsigned& shhdr) const;`
C API	`H5Pget_version`
Exception	`PropListIException` if H5Pget_version returns a negative value

4.14.5. setUserblock

Prototype	`void setUserblock(hsize_t size) const;`
C API	`H5Pset_userblock`
Exception	`PropListIException` if `H5Pset_userblock` returns a negative value

4.14.6. getUserblock

Prototype	`hsize_t getUserblock() const;`
C API	`H5Pget_userblock`
Description	Return the size of a user block obtained from the function H5Pget_userblock, if it is a non-negative value.
Exception	`PropListIException` if H5Pget_userblock returns a negative value

4.14.7. setSizes

Prototype	`void setSizes(size_t sizeof_addr = 4, size_t sizeof_size = 4) const;`
C API	`H5Pset_sizes`
Exception	`PropListIException` if H5Pset_sizes returns a negative value

4.14.8. getSizes

Prototype	`void getSizes(size_t& sizeof_addr, size_t& sizeof_size) const;`
C API	`H5Pget_sizes`
Exception	`PropListIException` if H5Pget_sizes returns a negative value

4.14.9. setSymk

Prototype	`void setSymk(unsigned int_nodes_k, unsigned leaf_nodes_k) const;`

C API	H5Pset_sym_k
Exception	PropListIException if H5Pset_sym_k returns a negative value

4.14.10. getSymk

Prototype	void getSymk(unsigned& int_nodes_k, unsigned& leaf_nodes_k) const;
C API	H5Pget_sym_k
Exception	PropListIException if H5Pset_sym_k returns a negative value

4.14.11. setIstorek

Prototype	void setIstorek(unsigned ik) const;
C API	H5Pset_istore_k
Exception	PropListIException if H5Pset_istore_k returns a negative value

4.14.12. getIstorek

Prototype	unsigned getIstorek() const;
C API	H5Pget_istore_k
Description	Return the rank obtained from the function H5Pget_istore_k, if it is a non-negative value.
Exception	PropListIException if H5Pget_istore_k returns negative value

4.14.13. ~FileCreatPropList virtual destructor

Prototype	virtual ~ FileCreatPropList ()
Description	No operation

4.15. FloatType

4.15.1. FloatType default constructor

Prototype	FloatType() : AtomType()

4.15.2. FLoatType copy constructor

Prototype	FloatType(const FloatType& original) : AtomType(original)
Description	No operation

4.15.3. FloatType constructor from a PredType

Prototype	FloatType(const PredType& pred_type) : AtomType();
Description	Invokes the member function copy to copy pred_type to this FloatType.

4.15.4. FloatType constructor from a DataSet

Prototype	FloatType(const DataSet& dataset);
C API	H5Dget_type
Exception	Exception if H5Dget_type returns a zero or a negative value

4.15.5. FloatType constructor using an existing id

Prototype	FloatType(const hid_t existing_id) : AtomType(existing_id)
Description	No operation

4.15.6. getFields

Prototype	`void getFields(size_t& spos, size_t& epos, size_t& esize, size_t& mpos, size_t& msize) const;`
C API	`H5Tget_fields`
Exception	`DataTypeIException` if `H5Tget_fields` returns a negative value

4.15.7. setFields

Prototype	`void setFields(size_t spos, size_t epos, size_t esize, size_t mpos, size_t msize) const;`
C API	`H5Tset_fields`
Exception	`DataTypeIException` if `H5Tset_fields` returns a negative value

4.15.8. getEbias

Prototype	`size_t getEbias() const;`
C API	`H5Tget_ebias`
Description	Return the exponent bias obtained from `H5Tget_ebias` if it is not 0.
Exception	`DataTypeIException` if `H5Tget_ebias` returns 0.

4.15.9. setEbias

Prototype	`void setEbias(size_t ebias) const;`
C API	`H5Tset_ebias`
Exception	`DataTypeIException` if `H5Tset_ebias` returns a negative value

4.15.10. getNorm

Prototype	`H5T_norm_t getNorm(string& norm_string) const;`
C API	`H5Tget_norm`
Description	Return the normalization type obtained from `H5Tget_norm` if it is not equal to `H5T_NORM_ERROR`.
Exception	`DataTypeIException` if `H5Tget_norm` returns H5T_NORM_ERROR

4.15.11. setNorm

Prototype	`void setNorm(H5T_norm_t norm) const;`
C API	`H5Tset_norm`
Exception	`DataTypeIException` if `H5Tset_norm` returns a negative value

4.15.12. getInpad

Prototype	`H5T_pad_t getInpad(string& pad_string) const;`
C API	`H5Tget_inpad`
Description	Return the padding type obtained from `H5Tget_inpad` if it is not equal to `H5T_PAD_ERROR`.
Exception	`DataTypeIException` if `H5Tget_inpad` returns `H5T_PAD_ERROR`

4.15.13. setInpad

Prototype	`void setInpad(H5T_pad_t inpad) const;`
C API	`H5Tset_inpad`
Exception	`DataTypeIException` if `H5Tset_inpad` returns a negative value

4.15.14. ~FloatType virtual destructor

Prototype	`virtual ~FloatType()`

Description	No operation

4.16. Group

4.16.1. Group default constructor

Prototype	`Group(): IdComponent();`
Description	No operation

4.16.2. Group copy constructor

Prototype	`Group(const H5Object& original) : IdComponent(original)`
Description	No operation

4.16.3. Group constructor using existing object id

Prototype	`Group(const hid_t object_id) : IdComponent(object_id)`
Description	No operation

4.16.4. getNumObjs

Prototype	`hsize_t getNumObjs() const;`
C API	`H5Gget_num_objs`
Exception	Invoke `throwException` if `H5Gget_num_objs` returns a negative value

4.16.5. getObjnameByIdx

Prototype	`ssize_t getObjnameByIdx(hsize_t idx, string& name, size_t size) const;`
C API	`H5Gget_objname_by_idx`
Exception	Invoke `throwException` if `H5Gget_objname_by_idx` returns a negative value

4.16.6. getObjTypeByIdx

Prototype	`int getObjTypeByIdx(hsize_t idx) const;` `int getObjTypeByIdx(hsize_t idx, string& type_name) const;`
C API	`H5Gget_objtype_by_idx`
Exception	Invoke `throwException` if `H5Gget_objtype_by_idx` returns the value `H5G_UNKNOWN`

4.16.7. getLocId

Prototype	`virtual hid_t getLocId() const;`
Description	This function simply calls `getId` to return the id of the group.

4.16.8. throwException

Prototype	`virtual void throwException(const string& func_name, const string& msg) const;`
Description	The first argument, `func_name`, is the name of the member function where an exception is to be raised.

	`throwException` will form a complete name by adding "Group::" to the front of `func_name`.
Exception	`GroupIException`

4.16.9. ~Group virtual destructor

Prototype	`virtual ~Group()`
Description	No operation

4.17. H5File

4.17.1. H5File default constructor

Prototype	`H5File() : IdComponent();`
Description	No operation

4.17.2. H5File copy constructor

Prototype	`H5File(const H5File& original) : IdComponent(original);`
Description	No operation

4.17.3. H5File constructor with file information

Prototype	`H5File(const string& name, unsigned int flags, const FileCreatPropList& create_plist = FileCreatPropList::DEFAULT, const FileAccPropList& access_plist = FileAccPropList::DEFAULT);`
	
	`H5File(const char* name, unsigned int flags, const FileCreatPropList& create_plist = FileCreatPropList::DEFAULT, const FileAccPropList& access_plist = FileAccPropList::DEFAULT);`

Description	These overloading constructors call `getFile` to open or to create an HDF5 file.

4.17.4. isHdf5

Prototype	`static bool isHdf5(const string& name);`
	
	`static bool isHdf5(const char* name);`

C API	`H5Fis_hdf5`
Description	If H5Fis_hdf5 returns a positive value, then return true, a zero, then return false, others, then raise an exception
Exception	`FileIException` if `H5Fis_hdf5` returns a negative value

4.17.5. reopen

Prototype	`void reopen();`
C API	`H5Freopen`
Exception	`FileIException` if `H5Freopen` returns a negative value

4.17.6. getCreatePlist

Prototype	`FileCreatPropList getCreatePlist() const;`
C API	`H5Fget_create_plist`
Description	If `H5Fget_create_plist` returns a positive value, create and return a FileCreatPropList instance.
Exception	`FileIException` if `H5Fget_create_plist` returns a a zero or a negative value

4.17.7. getAccessPlist

Prototype	`FileAccPropList getAccessPlist() const;`
C API	`H5Fget_access_plist`
Description	If `H5Fget_access_plist` returns a positive value, create and return a `FileAccPropList` instance.
Exception	`FileIException` if `H5Fget_access_plist` returns a zero or a negative value

4.17.8. throwException

Prototype	`virtual void throwException(const string& func_name, const string& msg) const;`
Description	The first argument, `func_name`, is the name of the member function where an exception is to be raised. `throwException` will form a complete name by adding "Group::" to the front of `func_name`.
Exception	`FileIException`

4.17.9. getFile

Prototype	`void getFile(const char* name, unsigned int flags, const FileCreatPropList& create_plist, const FileAccPropList& access_plist);`
C API	`H5Fcreate or H5Fopen`
Description	This function is private and is used by a constructor to create or open an HDF5 file. if `flags` & (`H5F_ACC_EXCL`|`H5F_ACC_TRUNC`|`H5F_ACC_DEBUG`), then `getFile` will invoke `H5Fcreate`, otherwise, `H5Fopen`.
Exception	`FileIException` if `H5Fcreate` or `H5Fopen` returns a zero or a negative value.

4.17.10. ~H5File virtual destructor

Prototype	`virtual ~H5File()`
Description	No operation

4.18. H5Library

All the member functions in this class are static so they can be invoked without an instance of H5Library.

4.18.1. open

Prototype	`static void open();`

C API	H5open
Exception	`LibraryIException` if `H5open` returns a negative value

4.18.2. close

Prototype	`static void close();`
C API	`H5close`
Exception	`LibraryIException` if `H5close` returns a negative value

4.18.3. dontAtExit

Prototype	`static void dontAtExit()`
C API	`H5dont_atexit`
Exception	`LibraryIException` if `H5dont_atexit` returns a negative value

4.18.4. getLibVersion

Prototype	`static void getLibVersion(unsigned& majnum, unsigned& minnum, unsigned& relnum)`
C API	`H5get_libversion`
Exception	`LibraryIException` if `H5get_libversion` returns a negative value

4.18.5. checkVersion

Prototype	`static void checkVersion(unsigned majnum, unsigned minnum, unsigned relnum)`
C API	`H5check_version`
Exception	`LibraryIException` if `H5check_version` returns a negative value

4.19. H5Object

4.19.1. H5Object default constructor

Prototype	`H5Object():IdComponent();`
Description	No operation

4.19.2. H5Object copy constructor

Prototype	`H5Object (const H5Object& original) : IdComponent(original)`
Description	No operation

4.19.3. H5Object constructor using existing object id

Prototype	`H5Object(const hid_t object_id) : IdComponent(object_id)`
Description	No operation

4.19.4. flush

Prototype	`void flush(H5F_scope_t scope) const`
C API	`H5Fflush`
Exception	`FileIException` if `H5Fflush` returns negative value

4.19.5. createAttribute

Prototype	`Attribute createAttribute(const char* name, const DataType& type, const DataSpace& space, const PropList& create_plist = PropList::DEFAULT) const;`
	
	`Attribute createAttribute(const string& name, const DataType& type, const DataSpace& space, const PropList& create_plist = PropList::DEFAULT) const;`

C API	`H5Acreate`
Description	If `H5Acreate` returns a positive value, then create and return an `Attribute` instance.
Exception	`AttributeIException` if `H5Acreate` returns a zero or a negative value

4.19.6. openAttribute using attribute name

Prototype	`Attribute openAttribute(const string& name) const;`
	
	`Attribute openAttribute(const char* name) const;`

C API	`H5Aopen_name`
Description	If `H5Acreate` returns a positive value, then create and return an `Attribute` instance. The `openAttribute` function that takes a `string` invokes the `openAttribute` function that takes a `char` pointer.
Exception	`AttributeIException` if `H5Aopen_name` returns a zero or a negative value

4.19.7. openAttribute using attribute index

Prototype	`Attribute openAttribute(const unsigned int idx) const;`
C API	`H5Aopen_idx`
Description	This overloading function takes the index of an attribute. If `H5Aopen_idx` returns a positive value, then create and return an `Attribute`.
Exception	`AttributeIException` if `H5Aopen_idx` returns a zero or a negative value

4.19.8. iterateAttrs

Prototype	`int iterateAttrs(attr_operator_t user_op, unsigned* idx = NULL, void* op_data = NULL)`
C API	`H5Aiterate`

Description	The following internal class is used to pass user's function and data to H5Aiterate.
	
	// user data for attribute iteration
	`class UserData4Aiterate {`

	public: unsigned int* idx; attr_operator_t op; void* opData; H5Object* object; }; The operator function is defined as: typedef void (*attr_operator_t)(H5Object& loc/*in*/, const string attr_name/*in*/, void *operator_data /*in,out*/);
Exception	AttributeIException if H5Aiterate returns negative value

4.19.9. getNumAttrs

Prototype	int getNumAttrs() const;
C API	H5Aget_num_attrs
Description	Return the number of attributes from H5Aget_num_attrs if it is a non-negative value
Exception	AttributeIException if H5Aget_num_attrs returns a negative value

4.19.10. removeAttr

Prototype	void removeAttr(const string& name) const;
	
	void removeAttr(const char* name) const;

C API	H5Adelete
Description	The function removeAttr that takes a string for name invokes the one that takes a char pointer.
Exception	AttributeIException if H5Adelete returns a negative value

4.19.11. ~H5Object virtual destructor

Prototype	virtual ~H5Object()
Description	No operation

4.20. IdComponent

This section is incomplete.

4.20.1. IdComponent copy constructor

Prototype	IdComponent(const IdComponent& original)

4.20.2. Constructor that takes an HDF5 identifier

Prototype	IdComponent(const hid_t h5_id)

4.20.3. getId

Prototype	`virtual hid_t getId () const`
Description	Returns the value of the instance' member `id`

4.20.4. setId

Prototype	`void setId(hid_t new_id)`
Description	Sets the instance' member `id` to `new_id`.

4.20.5. incRefCount

Prototype	`void incRefCount()`
C API	
Description	
Exception	`IdComponentException` if `Cfunction` returns negative value

Increment reference counter

4.20.6. decRefCount

Prototype	`void decRefCount()`
C API	
Description	
Exception	`IdComponentException` if `Cfunction` returns negative value

Decrement reference counter

4.20.7. getCounter

Prototype	`int getCounter()`
C API	
Description	
Exception	`IdComponentException` if `Cfunction` returns negative value

Gets the reference counter to this identifier

4.20.8. noReference

Prototype	`bool noReference()`
C API	
Description	
Exception	`IdComponentException` if `Cfunction` returns negative value

Determines whether there is no more reference to this identifier; note: the reference counter is decremented before checking

4.20.9. operator=

Prototype	`IdComponent& operator=(const IdComponent& rhs)`
C API	
Description	
Exception	`IdComponentException` if `Cfunction` returns negative value

Determines whether there is no more reference to this identifier; note: the reference counter is decremented before checking

4.20.10. reset

Prototype	Reset - obsolete

4.20.11. ~IdComponent virtual destructor

Prototype	`virtual ~IdComponent()`
Description	No operation

4.21. IntType

4.21.1. IntType default constructor

Prototype	`IntType() : AtomType()`

4.21.2. IntType copy constructor

Prototype	`IntType(const IntType& original) : AtomType (original)`
Description	No operation

4.21.3. IntType constructor from a PredType

Prototype	`IntType(const PredType& pred_type) : AtomType();`
Description	Invoke the member function `copy` to copy `pred_type` to this `IntType`.

4.21.4. IntType constructor from a DataSet

Prototype	`IntType (const DataSet& dataset);`
C API	`H5Dget_type`
Exception	`DataTypeIException` if `H5Dget_type` returns a zero or a negative value.

4.21.5. IntType constructor using an existing id

Prototype	`IntType(const hid_t existing_id) : AtomType(existing_id)`
Description	No operation

4.21.6. getSign

Prototype	`H5T_sign_t getSign() const;`
C API	`H5Tget_sign`
Description	Return the sign type obtained from `H5Tget_sign` if it is not H5T_SGN_ERROR.
Exception	`DataTypeIException` if `H5Tget_sign` returns `H5T_SGN_ERROR`

4.21.7. setSign

Prototype	`void setSign(H5T_sign_t sign) const;`
C API	`H5Tset_sign`
Exception	`DataTypeIException` if `H5Tset_sign` returns a negative value

4.21.8. ~IntType virtual destructor

Prototype	`virtual ~IntType()`
Description	No operation

4.22. PredType

This class contains the definition of objects that correspond to the predefined datatypes defined in the HDF5 library. Refer to Appendix A specific names.

4.22.1. PredType default constructor

Prototype	`PredType() : AtomType()`
Description	No operation

4.22.2. PredType copy constructor

Prototype	`PredType(const PredType& original) : AtomType(original)`
Description	No operation

4.22.3. PredType constructor using an existing id

Prototype	`PredType(const hid_t predtype_id) : AtomType(predtype_id)`
Description	Set data member `is_predtype` to `true` because this is a predefined datatype.

4.22.4. getId

Prototype	`virtual hid_t getId() const;`
Description	Use switch on the data member `id` to determine the appropriate HDF5 defined type to return. Return `INVALID` if `id` is not one of the enum values.

4.22.5. operator=

Prototype	`PredType& operator= (const PredType& rhs);`
Description	Invoke the member function `copy` to copy the `rhs` to `this`, then return `*this`.

4.22.6. ~PredType virtual destructor

Prototype	`virtual ~PredType();`
Description	No operation

4.23. PropList

4.23.1. PropList default constructor

Prototype	`PropList(): IdComponent(0)`
Description	No operation

4.23.2. PropList copy constructor

Prototype	`PropList(const PropList& original) : IdComponent(original)`
Description	No operation

4.23.3. PropList constructor using an existing id

Prototype	`PropList(const hid_t plist_id);`
C API	`H5Pcreate`

Description	Use H5Iget_type to determine whether or not plist_id is of type H5I_GENPROP_CLS. If it is, invoke H5Pcreate on plist_id to create the property list. If plist_id is equal to H5P_NO_CLASS, set id to H5P_DEFAULT, otherwise, to plist_id.
Exception	`PropListIException` if `H5Pcreate` returns a zero or a negative value

4.23.4. copy

Prototype	`void copy(const PropList& like_plist);`
C API	`H5Pcopy`
Description	Reset the identifier of this instance before copying `like_plist` to it, to make certain that the previous property list is properly closed.
Exception	`PropListIException` if `H5Pcopy` returns a zero or a negative value

4.23.5. operator=

Prototype	`PropList& operator=(const PropList& rhs);`
Description	Invoke member function `copy` to copy `rhs` to `this`, then return `*this`.

4.23.6. copyProp

Prototype	`void copyProp(PropList& dest, PropList& src, const string& name);`
	
	`void copyProp(PropList& dest, PropList& src, const char* name);`

C API	`H5Pcopy_prop`
Description	The function `copyProp` that takes a string for name invokes the one that takes a char pointer.
Exception	`PropListIException` if `H5Pcopy_prop` returns a negative value

4.23.7. getClass

Prototype	`hid_t getClass() const;`
C API	`H5Pget_class`
Description	Return the class obtained from the function H5Pget_class, if it is not equal to H5P_NO_CLASS.
Exception	`PropListIException` if `H5Pget_class` returns H5P_NO_CLASS

4.23.8. p_close

Prototype	`virtual void p_close() const;`
Description	This function is removed due to the recent availability of the reference counting mechanism in the HDF5 library.

4.23.9. ~PropList virtual destructor

Prototype	`virtual ~PropList()`
Description	No operation

4.24. RefCounter

This class will be removed in future releases, thus, it is unnecessary to list its members. However, keeping this subsection here makes the section numbering consistent with the previous section, Class Description.

4.25. StrType

4.25.1. StrType default constructor

Prototype	`StrType(): AtomType()`
Description	No operation

4.25.2. StrType copy constructor

Prototype	`StrType(const StrType& original) : AtomType (original)`
Description	No operation

4.25.3. StrType constructor from a PredType

Prototype	`StrType(const PredType& pred_type) : AtomType();`
Description	Invokes the member function `copy` to copy `pred_type` to this `IntType`.

4.25.4. StrType constructor from a PredType and with specified length

Prototype	`StrType(const PredType& pred_type, const size_t size) : AtomType();`
Description	Invokes the member function `copy` to copy `pred_type` to this `StrType`, and the member function `setSize` to set the string to the specified length, size.

4.25.5. StrType constructor from a DataSet

Prototype	`StrType(const DataSet& dataset);`
C API	`H5Dget_type`
Exception	`DataTypeIException` if `H5Tget_type` returns a zero or a negative value

4.25.6. StrType constructor using an existing id

Prototype	`StrType(const hid_t existing_id) : AtomType(existing_id)`
Description	No operation

4.25.7. getCset

Prototype	`H5T_cset_t getCset() const;`
C API	`H5Tget_cset`
Description	Return the character set type obtained from `H5Tget_cset` if it is not equal to `H5T_CSET_ERROR`.

Exception	`DataTypeIException` if `H5Tget_cset` returns `H5T_CSET_ERROR`

4.25.8. setCset

Prototype	`void setCset(H5T_cset_t cset) const;`
C API	`H5Tset_cset`
Exception	`DataTypeIException` if `H5Tset_cset` returns a negative value

4.25.9. getStrpad

Prototype	`H5T_str_t getStrpad() const;`
C API	`H5Tget_strpad`
Description	Return the string padding method obtained from `H5Tget_strpad` if it is not equal to `H5T_STR_ERROR`.
Exception	`DataTypeIException` if `H5Tget_strpad` returns `H5T_STR_ERROR`

4.25.10. setStrpad

Prototype	`void setStrpad(H5T_str_t strpad) const;`
C API	`H5Tset_strpad`
Exception	`DataTypeIException` if `H5Tset_strpad` returns a negative value

4.25.11. ~StrType virtual destructor

Prototype	`virtual ~StrType()`
Description	No operation

5. Data Dictionary

This section lists all members of any classes and their types and description.

5.1. Exception

detail_message	string; holds a descriptive message of the exception
func_name	string; holds the name of the function from which the exception is thrown

5.2. IdComponent

id	hid_t

5.3. PredType

predefined_types	Enumeration used to determine PredType id from HDF5 predefined types.

6. Analysis and Design Rationale

This section describes any decision regarding any design issues encountered or any coding rules throughout the API.

6.1. Encapsulation of the HDF5 id

The HDF5 library uses identifiers (hid_t id) in specifying objects. A decision has been made that the C++ API will encapsulate the id in classes.

6.2. Exception rules

Everytime an exception is thrown, it should have the member function name for the first parameter, and a message indicating that the C API failed for the second parameter.

6.3. Reference Counting Mechanism

Originally, the C++ API handled object reference counting at the wrapper level via the class RefCounter. When the HDF5 C library started providing the reference counting mechanism at its level, the C++ APIs was revised to utilize the new APIs. As the result, class RefCounter is removed and class IdComponent provides the wrappers for these new C APIs.

7. Current Status

7.1. Coding

7.1.1. Not yet implemented:

BitFieldType: is a user-defined bitfield datatype and is not yet implemented.

OpaqueType: is a user-defined opaque datatype and is not yet implemented.

7.1.2. 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.

7.1.3. Most C API iterate functions are not yet implemented.

7.1.4.

7.2. Testing

In addition to test cases that are specific to the C++ API, the C++ tests are implemented closely following the C tests. Currently, the following test sets are done:

- dsets.cpp: tests dataset basic functionality
- tfile.cpp: tests file basic functionality
- th5s.cpp: tests basic dataspace

7.3. Documentation

In addition to this document, which specifies to the developers how to implement the C++ API, a Reference Manual is available to the users. An online version of the RM can be found at: (Frank?) A PDF file will be available later.

It is also in the plan that a Tutorial will be provided online and similar to the available C Tutorial.

8. Future Works

Missing Basic Functionalities

Add list of which wrappers are not done.

Planned Features

Add list of planned functions.

- In the later versions of the C++ API, class DataSpace may be broken into a class hierarchy that reflects the nature of the HDF5 dataspace.
- In classes Attribute and DataSet, read and write may be implemented using operators >> and << in the later versions of the C++ API.
- In various classes, short-cut functions may be implemented for user's convenience.

User Requested Features

Add list of user requested functions.

Glossary

To do list for this doc:

- Make sure class names, function names, var names have appropriate format, in tables too
- fix diagrams
- complete section 8, Future Works
- add missing functions to Section 4
- make sure all private functions are included in both Sections 3 and 4
- label all private/protected functions
- add appendices for predefined type and any other