

Release Notes

General Comments

This distribution provides .NET wrappers for several HDF5 functions. A complete list of the implemented wrappers is included at the end of this document.

While the wrapper API mirrors that of the underlying "C" library there are some obvious differences. First, the wrappers are implemented as static members of classes named for each of the "C" API interfaces. Thus the "C" function `H5Fcreate()` is wrapped as `H5F.create()`. A second difference is the addition of data types to provide a type-safe interface. In most instances the `hid_t` type has been replaced with a data type that is specific to the id function. These data types include such types as `H5FileId`, `H5GroupId`, and `H5DataSetId`. The use of these types provide compile-time checks to ensure appropriate id usage. Finally, rather than return status codes, the wrappers throw exceptions when requested operations fail.

In general, function overloading has been used to replace functions that can take null id's. For example, rather than allow the last parameter of `H5F.create` to be null, the overload functions:

```
H5F.create(string filename, CreateMode mode)
```

```
H5F.create(string filename, CreateMode mode, H5PropertyListId id)
```

are provided instead.

It is always difficult to pick variable, datatype, and class names that make sense and satisfy diverse concepts of aesthetics. We've done our best to pick good names, but we are happy to consider other ideas at this point when such changes can be made with relatively little effort.

This release contains the following files (indentation reflects directory structure):

.\HDF5DotNet directory

 \Examples - Example files in 3 languages

 \CSharpExample1 - C# example

 Program.cs - source code for C# example

 CSharpExample1.sln - Visual Studio solution file

 \cppExample - C++\CLI example

 example1.cpp - source code for C++ example

 cppExample.sln - Visual Studio solution file

 \tests - test suite for the .NET API

 \HDF5DotNet - files to build HDF5DotNet.dll

 makeEx.cpp - source file for makeEx.exe, exception class generator.

 makeEx.exe - exception class generator. This program writes the

 source code for HDFExceptionSubclasses.h and
 HDFExceptionSubclasses.cpp. It is strictly a

 developer tool.

 exceptionNames - input file for makeEx.exe

 makeDoc.bat - script that uses Sandcastle (alpha) documentation

 tool to produce help files. This script will not be

 needed soon as the graphical user interface for

 sandcastle becomes available. At present it assumes

 that Sandcastle is installed in its

 default location ("C:\Program Files\Sandcastle").

 Sandcastle is available

 from <http://blogs.msdn.com/sandcastle/>

Installation of Sandcastle is only necessary if you are making modifications to the HDF5DotNet help files.

H5*.h and H5*.cpp - source files for HDF5 wrapper classes.

These classes contain static functions wrap HDF5 native functions.

\doc\output

HDF5DotNet.chm - help files. Double click this file to see the VisualStudio-style library documentation.

\doc\output\html

HTML versions of the help files. These seem to be missing a link from the class to the member functions, but the html files for the members are available in this directory.

ReleaseNotes.pdf - this file.

\include - HDF5 include files with some minor modifications.

To build a HDF5DotNet C# Project:

1.) Make sure that the files hdf5dll.dll, hdf5dll.lib, and szlibdll.dll are installed in \Windows\System32 or in the project's working directory.

You can obtain szlibdll.dll from:

<ftp://ftp.ncsa.uiuc.edu/HDF/lib-external/szip/2.0/bin/windows/>

HDF5 uses Szip version 2.0 for compression and Szip compression software is provided with HDF5 products starting from 1.6.0 release. To use Szip 2.0 library, you can download Szip source codes and binaries from <ftp://hdf.ncsa.uiuc.edu/lib-external/szip/bin/windows>

Please note that Szip is not a totally open-source free software. For licensing issue of Szip, please check "Licensing terms" at http://hdf.ncsa.uiuc.edu/doc_resource/SZIP/index.html.

You can obtain hd5fdll.dll and hd5fdll.lib from:

- 2.) Start Microsoft Visual Studio 2005. Create a C# project.
- 3.) Select Project | Add Reference from Visual Studio's Menu.
Select the Browse Tab.
Browse to HDF5DotNet\HDF5DotNet\Debug and select HDF5DotNet.dll (press OK).
- 4.) Add the statement "using HDF5DotNet;" to the C# source file.
- 5.) You should be able to write your program now utilizing the HDF5 "IntelliSense". Type "H5F." and you should see the available completions.

Documentation for the implemented functions is available through the help files at HDF5DotNet\HDF5DotNet\doc\output\HDF5DotNet.chm. Double-click this file to browse the help files. The "Examples" directory provides sample projects in C#, C++/CLI, and Visual Basic

Because these functions are wrappers, their functions are similar to those described for the HDF5 "C" library. The following table provides a list of implemented functions and shows their relationships to the the "C" library functions.

- H5F

```
hid_t H5Fcreate (const char *filename, unsigned flags, hid_t
create_plist, hid_t access_plist);
```

```
public static H5FileId create(string filename, H5F.CreateMode mode);
public static H5FileId create(string filename, H5F.CreateMode mode,
H5PropertyListId accessPropertyList);
public static H5FileId create(string filename, H5F.CreateMode mode,
H5PropertyListId creationPropertyList, H5PropertyListId
accessPropertyList);
```

Comments: The overloads replace the use of null parameters.

```
hid_t H5Fopen (const char *filename, unsigned flags, hid_t
access_plist);
```

```
public static H5FileId open(string filename, H5F.OpenMode mode);
public static H5FileId open(string filename, H5F.OpenMode mode,
H5PropertyListId propertyListId);
```

```
herr_t H5Fclose (hid_t file_id);
```

```
public static void close(H5FileId id);
```

- H5S

```
hid_t H5Screate(H5S_class_t type);
```

```
public static H5DataSpaceId create(H5S.H5SClass createClass);
```

```
hid_t H5Screate_simple(int rank, const hsize_t dims[], const hsize_t maxdims[]);
```

```
public static H5DataSpaceId create_simple(int rank, ulong[] dims);  
public static H5DataSpaceId create_simple(int rank, ulong[] dims,  
ulong[] maxDims);
```

```
herr_t H5Sselect_hyperslab(hid_t space_id, H5S_seloper_t op, const  
hsize_t start[], const hsize_t _stride[], const hsize_t count[], const  
hsize_t _block[]);
```

```
public static void selectHyperslab(H5DataSpaceId spaceId,  
H5S.SelectOperator selectOperator, ulong[] start, ulong[] count);  
public static void selectHyperslab(H5DataSpaceId spaceId,  
H5S.SelectOperator selectOperator, ulong[] start, ulong[] count,  
ulong[] block);  
public static void selectStridedHyperslab(H5DataSpaceId spaceId,  
H5S.SelectOperator selectOperator, ulong[] start, ulong[] stride,  
ulong[] count);  
public static void selectStridedHyperslab(H5DataSpaceId spaceId,  
H5S.SelectOperator selectOperator, ulong[] start, ulong[] stride,  
ulong[] count, ulong[] block);
```

Comments: selectStridedHypderslab was added to disambiguate function calls with null (missing) parameters.

```
Int H5Sget_simple_extent_ndims(hid_t space_id);
```

```
public static int getSimpleExtentNDims(H5DataSpaceId spaceId);
```

```
int H5Sget_simple_extent_dims(hid_t space_id, hsize_t dims[], hsize_t maxdims[]);
```

```
public static ulong[] getSimpleExtentDims(H5DataSpaceId spaceId);  
public static ulong[] getSimpleExtentMaxDims(H5DataSpaceId spaceId);
```

Comments: The two new functions return arrays containing either current or maximum dimensions.

```
herr_t H5Sselect_none(hid_t space_id)
```

```
public static bool selectNone(H5DataSpaceId spaceId);
```

```
herr_t H5Sclose(hid_t dspace);
```

```
public static void close(H5DataSpaceId id);
```

- H5T

```
hid_t H5Topen(hid_t loc_id, const char *name);
```

```
public static H5DataTypeId open(H5LocId groupOrFileId, string  
datatypeName);
```

```
hid_t H5Tcreate(H5T_class_t type, size_t size);
```

```
public static H5DataTypeId create(H5T.CreateClass createClass, uint  
size);
```

```
hid_t H5Tcopy(hid_t type_id);
```

```
public static H5DataTypeId copy(H5DataSetId dataSetId);
```

```
public static H5DataTypeId copy(H5DataTypeId typeId);
```

```
public static H5DataTypeId copy(H5T.H5Type stdType);
```

```
herr_t H5Tcommit(hid_t loc_id, const char * name, hid_t type )
```

```
public static void commit(H5LocId location, string dataTypeName,  
H5DataTypeId typeId);
```

```
herr_t H5Tclose(hid_t type_id);
```

```
public static void close(H5DataTypeId typeId);
```

```
herr_t H5Tinsert(hid_t parent_id, const char *name, size_t offset,  
hid_t member_id); - for all types in H5T_class_t
```

```
public static void insert(H5DataTypeId compoundDataType, string  
fieldName, uint offset, H5T.H5Type fieldId);  
  
public static void insert(H5DataTypeId compoundDataType, string  
fieldName, uint offset, H5DataTypeId fieldId);
```

```
hid_t H5Tenum_create(hid_t base_id);
```

```
public static H5DataTypeId enumCreate(H5DataTypeId parentId);  
public static H5DataTypeId enumCreate(H5T.H5Type h5Type);
```

```
herr_t H5Tenum_insert(hid_t type, const char *name, const void  
*value);
```

```
public static void enumInsert<Type>(H5DataTypeId typeId, string name,  
ref Type value);
```

```
hid_t H5Tget_native_type(hid_t type_id, H5T_direction_t direction);
```

```
public static H5DataTypeId getNativeType(H5DataTypeId typeId,  
H5T.Direction direction);  
  
public static H5DataTypeId getNativeType(H5T.H5Type h5Type,  
H5T.Direction direction);
```

```
hid_t H5Tvlen_create(hid_t base_id);
```

```
public static H5DataTypeId vlenCreate(H5DataTypeId baseId);  
public static H5DataTypeId vlenCreate(H5T.H5Type h5Type);
```

```
H5T_class_t H5Tget_class(hid_t type_id);
```

```
public static H5T.H5TClass getClass(H5DataTypeId typeId);  
public static H5T.H5Tclass getClass(H5T.H5Type h5Type);
```



```
size_t H5Tget_size(hid_t type_id);
```

```
public static void setSize(H5DataTypeId typeId, uint size);  
public static uint getSize(H5DataTypeId typeId);  
public static uint getSize(H5T.H5Type h5Type);
```

```
H5T_sign_t H5Tget_sign(hid_t type_id);
```

```
public static H5T.Sign getSign(H5DataTypeId typeId);  
public static H5T.Sign getSign(H5T.H5Type h5Type);
```

```
int H5Tget_nmembers(hid_t type_id);
```

```
public static int getNMembers(H5DataTypeId typeId);
```

```
char *H5Tget_member_name(hid_t type_id, unsigned membno);
```

```
public static string getMemberName(H5DataTypeId typeId, uint  
fieldIndex);
```

```
int H5Tget_member_index(hid_t type_id, const char *name);
```

```
public static int getMemberIndex(H5DataTypeId typeId, string  
fieldName);
```

```
H5T_class_t H5Tget_member_class(hid_t type_id, unsigned membno);
```

```
public static H5T.H5TClass getMemberClass(H5DataTypeId typeId, uint  
memberNumber);
```

- H5D

```
hid_t H5Dcreate(hid_t file_id, const char *name, hid_t type_id, hid_t
space_id, hid_t plist_id);
```

```
public static H5DataSetId create(H5LocId groupOrFileId, string
datasetName, H5DataTypeId dataTypeId, H5DataSpaceId dataspaceId);
public static H5DataSetId create(H5LocId groupOrFileId, string
datasetName, H5T.H5Type dataType, H5DataSpaceId dataspaceId);
```

```
hid_t H5Dopen(hid_t file_id, const char *name);
```

```
public static H5DataSetId open(H5LocId groupOrFileId, string
dataSetName);
```

```
herr_t H5Dclose(hid_t dset_id);
```

```
public static void close(H5DataSetId id);
```

```
herr_t H5Dread(hid_t dset_id, hid_t mem_type_id, hid_t mem_space_id,
hid_t file_space_id, hid_t plist_id, void *buf/*out*/);
```

```
public static void read<Type>(H5DataSetId dataSetId, H5DataTypeId
memTypeId, H5DataSpaceId memSpaceId, H5DataSpaceId fileSpaceId,
H5PropertyListId xferPropListId, H5Array<Type> data);
```

```
public static void read<Type>(H5DataSetId dataSetId, H5DataTypeId
memTypeId, H5DataSpaceId memSpaceId, H5DataSpaceId fileSpaceId,
H5PropertyListId xferPropListId, ref Type data);
```

```
public static void readScalar<Type>(H5DataSetId dataSetId,
H5DataTypeId dataType, H5Array<Type> data);
```

```
public static void readScalar<Type>(H5DataSetId dataSetId,
H5DataTypeId dataType, ref Type data);
```

Comment: The read functions read arrays, and the readScalar functions read single values (e.g., one int, float, char, or structures). All read and write functions require that the data be .NET value types, rather than reference types.

H5Array constructors can take any array of value types as a parameter. The array data is not copied. H5Array simply stores information about the array so that it can be passed to other routines without rank being explicitly specified.

```
herr_t H5Dwrite(hid_t dset_id, hid_t mem_type_id, hid_t mem_space_id,
hid_t file_space_id, hid_t plist_id, const void *buf);
```

```
public static void write<Type>(H5DataSetId dataSetId, H5DataTypeId
dataType, H5Array<Type> data);
```

```
public static void write<Type>(H5DataSetId dataSetId, H5DataTypeId
memTypeId, H5DataSpaceId memSpaceId, H5DataSpaceId fileSpaceId,
H5PropertyListId xferPropListId, H5Array<Type> data);
```

```
public static void writeScalar<Type>(H5DataSetId dataSetId,
H5DataTypeId memTypeId, H5DataSpaceId memSpaceId, H5DataSpaceId
fileSpaceId, H5PropertyListId xferPropListId, ref Type data);
```

```
public static void writeScalar<Type>(H5DataSetId dataSetId,
H5DataTypeId dataType, ref Type data);
```

Comment: The write functions write arrays, and the writeScalar functions write single values (e.g., one int, float, char, or structures). All read and write functions require that the data be .NET value types, rather than reference types.

H5Array constructors can take any array of value types as a parameter. The array data is not copied. H5Array simply stores information about the array so that it can be passed to other routines without rank being explicitly specified

```
hid_t H5Dget_space(hid_t dset_id);
```

```
public static H5DataSpaceId getSpace(H5DataSetId dataSetId);
```

```
hid_t H5Dget_type(hid_t dset_id);
```

```
public static H5DataTypeId getType(H5DataSetId dataSetId);
```

- H5G

```
hid_t H5Gcreate(hid_t loc_id, const char *name, size_t size_hint);
```

```
public static H5GroupId create(H5LocId fileId, string groupName, uint
sizeHint);
```

```
hid_t H5Gopen(hid_t loc_id, const char *name);
```

```
public static H5GroupId open(H5LocId groupOrFileId, string groupName);
```

```
herr_t H5Gget_objinfo(hid_t loc_id, const char *name, hbool_t  
follow_link, H5G_stat_t *statbuf/*out*/);
```

```
ObjectInfo getObjectInfo(H5LocId loc, string name, bool followLink)
```

Comment: ObjectInfo contains properties that provided access to the object information.

```
herr_t H5Giterate(hid_t loc_id, const char *name, int *idx_p,  
H5G_iterate_t op, void *op_data);
```

```
public static int iterate(H5LocId loc, string name, H5GIterateDelegate  
func, object parameters, int startIndex);
```

Comment: uses a delegate rather than a function pointer.

```
herr_t H5Gclose(hid_t group_id)
```

```
public static void close(H5GroupId groupId);
```

```
herr_t H5Gget_num_objs(hid_t loc_id, hsize_t* num_obj)
```

```
public static ulong getNumObjects(H5GroupId groupId);
```

```
ssize_t H5Gget_objname_by_idx(hid_t loc_id, hsize_t idx, char *name,  
size_t size);
```

```
public static string getObjectByNameIndex(H5GroupId groupId, int  
objectIndex);
```

- H5E

```
set_auto(NULL, NULL)
```

```
public static void suppressPrinting();
```

While you can use the HDF5DotNet.dll as downloaded to write new HDF5 projects, if you want to change the library itself you will need to recompile it. A Visual Studio solution file has been provided in HDF5DotNet\HDF5DotNet\HDF5DotNet.sln. The solution file uses relative paths for source files, but an absolute path for the hdf5 lib file. If hdf5dll.lib is placed in the \windows\system32 directory, the project should compile without modification. If you want to place hdf5dll.lib somewhere else, chose Project | Properties from the Visual Studio menu, and then under Configuration Properties/Linker/Input change the location of hdf5dll.lib in the "Additinal Dependencies" field to the desired location.

Solution files also exist in the Example subdirectories for a C# example (Examples\CSharpExample1\CSharpExample1.sln), and C++\CLI example (Examples\cppExample1\cppExample1.sln), and a Visual Basic example (Examples\VBexample1\VBExample1.sln). These can be compiled and run without need for modification from Visual Studio.

Please send any questions or comments to help at hdfgroup dot org.