Copying Committed Datatypes with H5Ocopy

HDF5 Release 1.8.9

May 2012

The HDF Group

http://www.HDFGroup.org

Copyright Notice and License Terms for HDF5 (Hierarchical Data Format 5) Software Library and Utilities

Redistribution and use in source and binary forms, with or without modification, are permitted for any purpose (including commercial purposes) provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the following disclaimer in the documentation and/or materials provided with the distribution.
3. In addition, redistributions of modified forms of the source or binary code must carry prominent notices stating that the original code was changed and the date of the change.
4. All publications or advertising materials mentioning features or use of this software are asked, but not required, to acknowledge that it was developed by The HDF Group and by the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign and credit the contributors.
5. Neither the name of The HDF Group, the name of the University, nor the name of any Contributor may be used to endorse or promote products derived from this software without specific prior written permission from The HDF Group, the University, or the Contributor, respectively.

DISCLAIMER: THIS SOFTWARE IS PROVIDED BY THE HDF GROUP AND THE CONTRIBUTORS "AS IS" WITH NO WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED. In no event shall The HDF Group or the Contributors be liable for any damages suffered by the users arising out of the use of this software, even if advised of the possibility of such damage.

Contributors: National Center for Supercomputing Applications (NCSA) at the University of Illinois, Fortner Software, Unidata Program Center (netCDF), The Independent JPEG Group (JPEG), Jean-loup Gailly and Mark Adler (gzip), and Digital Equipment Corporation (DEC).

Portions of HDF5 were developed with support from the Lawrence Berkeley National Laboratory (LBNL) and the United States Department of Energy under Prime Contract No. DE-AC02-05CH11231.

Portions of HDF5 were developed with support from the University of California, Lawrence Livermore National Laboratory (UC LLNL). The following statement applies to those portions of the product and must be retained in any redistribution of source code, binaries, documentation, and/or accompanying materials:

> This work was partially produced at the University of California, Lawrence Livermore National Laboratory (UC LLNL) under contract no. W-7405-ENG-48 (Contract 48) between the U.S. Department of Energy (DOE) and The Regents of the University of California (University) for the operation of UC LLNL.
>
> **DISCLAIMER:** This work was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately- owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

Contents

1. Copying Committed Datatypes with H5Ocopy 4
1.1. Callback Function 5
1.2. Function Summary 6
1.3. Resources 6

1. Copying Committed Datatypes with H5Ocopy

Committed datatypes can be a powerful feature in HDF5. They can be used to share a single datatype description among multiple datasets, to save space or ensure that the datatypes are truly identical, and to assign a name to that datatype within the HDF5 group structure. The object copy API, `H5Ocopy`, can be used to copy HDF5 objects from one file to another, including committed datatypes and objects that use them. However, problems can occur when a dataset using a committed datatype or an object with an attribute that uses a committed datatype is copied to another file with `H5Ocopy`.

When copying a dataset that uses a committed datatype or an object with an attribute that uses a committed datatype between files, the library by default does not look for a matching committed datatype in the destination file. The library creates a new committed datatype in the destination file without any links to it (an anonymous committed datatype) and then links the dataset to the anonymous committed datatype. This means that, when copying multiple datasets in separate calls to `H5Ocopy`, a new committed datatype is created for each `H5Ocopy` call. While it is possible to have all of the copied datasets share the same committed datatype by copying them in a single call to `H5Ocopy`, this is not always attainable.

For example, imagine that a user has an application that automatically creates many data files, each with many datasets that all use a single committed datatype. At the end of a project, the user wants to merge all of these files into a single file. The HDF5 Library can have all of the datasets in the combined file use the same committed datatype, but the default behavior of the library is to create an anonymous committed datatype for each dataset.

To make sure that shared committed datatypes in the source are shared in the copy, use the `H5Pset_copy_object` property list API routine to set the `H5O_COPY_MERGE_COMMITTED_DTYPE_FLAG` flag. When this flag is set and `H5Ocopy` encounters an object or attribute that uses a committed datatype, `H5Ocopy` will search for a matching committed datatype in the destination file. If a matching committed datatype is found, then it will be used by the copied dataset or attribute. The next few paragraphs describe in more detail the process that `H5Ocopy` goes through.

When the `H5O_COPY_MERGE_COMMITTED_DTYPE_FLAG` flag is set, `H5Ocopy` will search the destination file for committed datatypes and build a temporary list in memory of all the committed datatypes it finds. Then, whenever `H5Ocopy` encounters a dataset that uses a committed datatype or an object with an attribute that uses a committed datatype in the source, it will check that list to see if it contains a datatype equal to the source datatype. If `H5Ocopy` finds an equal datatype, it will modify the copied object or attribute to use the found committed datatype as its datatype. `H5Ocopy` will then update the list if a new committed datatype is created in the destination file as a result of the copy. When later datasets and attributes using committed datatypes are encountered, the library will again check to see if the list contains a matching datatype.

To determine if two committed datatypes are equal, the library will compare their descriptions in a manner similar to `H5Tequal`. In addition, if either committed datatype has one or more attributes, then all attributes must be present in both committed datatypes, and the attributes must all be identical. Each attribute's datatype description, dataspace, and raw data must be identical. However, if an attribute uses a committed datatype, then the attributes of the attribute's committed datatype will *not* be compared.

When `H5Ocopy` encounters a committed datatype object in the source file, it will similarly search for a matching committed datatype in the destination file. If a match is found, the library will create a hard link in

the destination file to the found datatype. If a match is not found, the library will copy the committed datatype normally and add it to the temporary list of committed datatypes in the destination file.

By default, `H5Ocopy` will search the entire destination file for a matching committed datatype. It is possible to focus where `H5Ocopy` will search. This focusing should result in a faster search. If there are locations in the destination file where a matching committed datatype might be found, then those locations can be specified with the `H5Padd_merge_committed_dtype_path` property.

The example below shows how to enable the feature described above for use with `H5Ocopy`.

```
hid_t ocpypl_id;

ocpypl_id = H5Pcreate(H5P_OBJECT_COPY);
status = H5Pset_copy_object(ocpypl_id, H5O_COPY_MERGE_COMMITTED_DT_FLAG);
status = H5Ocopy(file1_id, src_name, file2_id, dst_name, ocpypl_id,
H5P_DEFAULT);
```

Example 1. Setting the object copy property list

1.1. Callback Function

If no matching datatype is found in the locations specified by the call to `H5Padd_merge_committed_dtype_path`, then `H5Ocopy` will by default search the entire destination file. In some cases, this may not be desirable. For instance, the user may expect the datatype to always have a match in the specified locations and may wish to return an error if a match is not found. The user may also have a very large file for which the full search incurs a substantial performance penalty. In this instance, the user may wish to log these events so that other datatypes can be added with `H5Padd_merge_committed_dtype_path`, or the user may wish to abort the search and copy the datatype normally.

To support these use cases, the functions `H5Pset_mcdt_search_cb` and `H5Pget_mcdt_search_cb` have been added. These functions allow the user to define a callback function that will be called every time the list of paths added by `H5Padd_merge_committed_dtype_path` has been exhausted but before beginning the full search of the file. The prototype for the callback function is defined by `H5O_mcdt_search_cb_t`. The only argument to the callback function is a user supplied user data pointer, and the return value is an `enum`, defined by `H5O_mndt_search_ret_t`, which tells the library to either continue with the full file search, abort the search and copy the datatype normally (create a new committed datatype in the destination file), or return an error.

1.2. Function Summary

Functions used in committed datatype copying operations are listed below.

Function Listing 1. Committed datatype copying related functions

C Function Fortran	Purpose
H5Ocopy (none)	Allows an application to copy an object within an HDF5 file or to another HDF5 file.
H5Pset_copy_object h5pset_copy_object_f	Allows an application to set properties to be used when an object is copied.
H5Padd_merge_committed_dtype_path (none)	Allows an application to add a path to the list of paths that will be searched in the destination file for a matching committed datatype.
H5Pfree_merge_committed_dtype_paths (none)	Allows an application to clear the list of paths stored in the object copy property list ocpypl_id.
H5Pset_mcdt_search_cb (none)	Allows an application to set the callback function that H5Ocopy will invoke before searching the entire destination file for a matching committed datatype.
H5Pget_mcdt_search_cb (none)	Allows an application to retrieve the callback function from the specified object copy property list.
H5O_mcdt_search_cb_t (none)	Definition of the callback function set by H5Pset_mcdt_search_cb. Provides the mechanism by which a user application may set an action for H5Ocopy to take after checking all suggested paths for a matching committed datatype but before starting the global search of the destination file.

1.3. Resources

See the following for more information.

See the "HDF5 Datatypes" chapter in the *HDF5 User's Guide*.

See these entries in the *HDF5 Reference Manual*:

- H5Ocopy
- H5Pset_copy_object
- H5Padd_merge_committed_dtype_path
- H5Pfree_merge_committed_dtype_paths
- H5Pset_mcdt_search_cb
- H5Pget_mcdt_search_cb