MPI Atomicity

HDF5 Release 1.8.9

May 2012



<http://www.HDFGroup.org>

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

HDF5 (Hierarchical Data Format 5) Software Library and Utilities

Copyright 2006-2012 by The HDF Group.

NCSA HDF5 (Hierarchical Data Format 5) Software Library and Utilities

Copyright 1998-2006 by the Board of Trustees of the University of Illinois.

**All rights reserved.**

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.

??????? this copyright information needs review/editing. This text is a copy of the copyright statement for the file image operations doc.

Contents

1. Introduction to MPI Atomicity 5

1.1. MPI Atomicity Function Summary ??????? 5

1.2. Abbreviations ??????? 5

1.3. Developer Prerequisites ??????? 6

1.4. Resources ??????? 6

2. RM Entries 7

2.1. H5Fset\_mpi\_atomicity 7

2.2. H5Fget\_mpi\_atomicity 9

# Introduction to MPI Atomicity

???????

See HDFFV-7961.

## MPI Atomicity Function Summary ???????

Functions used in MPI atomicity operations are listed below.

| Function Listing 1. MPI atomicity functions | |
| --- | --- |
| C Function | Purpose |
| H5Pset\_file\_image | Allows an application to specify an initial file image. For more information, see page 12. |
| H5Pget\_file\_image | Allows an application to retrieve a copy of the file image designated for a VFD to use as the initial contents of a file. For more information, see page 12. |
| H5Pset\_file\_image\_callbacks | Allows an application to manage file image buffer allocation, copying, reallocation, and release. For more information, see page 13. |
| H5Pget\_file\_image\_callbacks | Allows an application to obtain the current file image callbacks from a file access property list. For more information, see page 16. |
| H5Fget\_file\_image | Provides a simple way to retrieve a copy of the image of an existing, open file. For more information, see page 18. |
| H5LTopen\_file\_image | Provides a convenient way to open an initial file image with the Core VFD. For more information, see page 19. |

## Abbreviations ???????

The following abbreviations are used in this document:

| Table 1. Abbreviations | |
| --- | --- |
| Abbreviation | This abbreviation is short for: |
| ??????? |  |
|  |  |
|  |  |

## Developer Prerequisites ???????

Developers who use the file image operations described in this document should be proficient and experienced users of the HDF5 C Library APIs. More specifically, developers should have a working knowledge of property lists, callbacks, and virtual file drivers.

## Resources ???????

See the following for more information.

The “RFC: File Image Operations” is the primary source for the information in this document.

The “Alternate File Storage Layouts and Low-level File Drivers” section is in “The HDF5 File” chapter of the *HDF5 User’s Guide* at <http://www.hdfgroup.org/HDF5/doc/UG/UG_frame08TheFile.html>.

The H5Pset\_fapl\_core function call can be used to modify the file access property list so that the Memory virtual file driver, H5FD\_CORE, is used. The Memory file driver is also known as the Core file driver. See the *HDF5 Reference Manual* at <http://www.hdfgroup.org/HDF5/doc/RM/RM_H5P.html#Property-SetFaplCore> for more information.

Links to the “Virtual File Layer” and “List of VFL Functions” documents can be found on the “HDF5 Technical Notes” page at <http://www.hdfgroup.org/HDF5/doc/TechNotes.html>.

# RM Entries

There are two MPI atomicity related APIs. They are H5Fset\_mpi\_atomicity and H5Fget\_mpi\_atomicity. The RM entries for the APIs appear below.

## H5Fset\_mpi\_atomicity

Name: H5Fset\_mpi\_atomicity

Signature:

herr\_t H5Fset\_mpi\_atomicity( hid\_t file\_id, hbool\_t flag )

Purpose:

Sets the atomicity mode to true, if flag is TRUE, or false if flag is FALSE.

Description:

The consistency semantics for data access using the file\_id is set by calling H5Fset\_mpi\_atomicity collectively among all processes belonging to the group of the communicator used to open the file. H5Fset\_mpi\_atomicity is collective; all processes have to pass the same value for file\_id and flag. This routine is only available if the HDF5 library is configured with parallel support (--enable-parallel) and is meant to be used when the only if the H5FD\_MPIO driver is used.

Limitations:

H5Fset\_mpi\_atomicity calls MPI\_File\_set\_atomicity underneath and is not supported if the execution platform does not support MPI\_File\_set\_atomicity. In the case where MPI\_File\_set\_atomicity is supported, the performance of data access operations might drop significantly is atomicity is set to true.

In certain scenarios, even if MPI\_File\_set\_atomicity is supported, setting atomicity to true using H5Fset\_mpi\_atomicity does not really yield atomic updates, because an H5Dwrite (for example) may translate into multiple MPI\_File\_write\_at functions underneath. This happens in all cases where the high level file access routine translates to multiple lower level file access routines. The following scenarios will encounter this issue:

Non-contiguous file access using independent I/O

Partial collective I/O using chunked access

Collective I/O using filters or when data conversion needs to happen

This means that atomicity is done per 1 MPI file access operation and not 1 HDF5 access operation, which is highly unlikely what most user wants. The use of barriers after the H5Dwrite and before the H5Dread in addition to setting atomicity to true in the user’s application would ensure additional ordering semantics and would actually solve this issue, since the barrier will guarantee that all underlying write operations will execute atomically before the read operations starts.

Parameters:

hid\_t file\_id

IN: Identifier of a file to terminate access to.

hbool\_t flag

IN: Logical flag to set the atomicity to.

Returns:

Returns a non-negative value if successful; otherwise returns a negative value.

Fortran90 Interface: h5fset\_mpi\_atomicity\_f

SUBROUTINE h5fset\_mpi\_atomicity\_f(file\_id, flag, hdferr)

IMPLICIT NONE

INTEGER(HID\_T), INTENT(IN) :: file\_id ! File identifier

INTEGER(HBOOL\_T), INTENT(IN) :: flag ! Atomicity flag

INTEGER, INTENT(OUT) :: hdferr ! Error code

! 0 on success and -1 on failure

END SUBROUTINE h5fset\_mpi\_atomicity\_f

## H5Fget\_mpi\_atomicity

Name: H5Fget\_mpi\_atomicity

Signature:

herr\_t H5Fget\_mpi\_atomicity( hid\_t file\_id, hbool\_t \*flag )

Purpose:

Returns the atomicity mode that is used.

Description:

H5Fget\_mpi\_atomicity returns the current consistency semantics mode for data access using the file\_id. If flag is true, then the atomic mode is enabled, otherwise, if flag is false, nonatomic mode is enabled.

Parameters:

hid\_t file\_id

IN: Identifier of a file to terminate access to.

hbool\_t \*flag

OUT: Logical flag to indicate atomic (true) or nonatomic (false) mode.

Returns:

Returns a non-negative value if successful; otherwise returns a negative value.

Fortran90 Interface: h5fget\_mpi\_atomicity\_f

SUBROUTINE h5fget\_mpi\_atomicity\_f(file\_id, flag, hdferr)

IMPLICIT NONE

INTEGER(HID\_T), INTENT(IN) :: file\_id ! File identifier

INTEGER(HBOOL\_T), INTENT(OUT) :: flag ! Atomicity flag

INTEGER, INTENT(OUT) :: hdferr ! Error code

! 0 on success and -1 on failure

END SUBROUTINE h5fget\_mpi\_atomicity\_f