HDF5 DAOS VOL Connector User's Guide

Neil Fortner, Jordan Henderson, Jerome Soumagne

This document aims to be a helpful guide on how to use the HDF5 DAOS VOL connector to leverage the capabilities of the DAOS object storage system within an HDF5 application.

Revision History

Version Number	Date	Comments
v0.1	Mar. 29, 2019	First draft.
v0.2	Apr. 24, 2019	Second draft.
v0.3	Aug. 27, 2019	Third draft.
v0.4	Oct. 25, 2019	Fourth draft.
v0.5	Nov. 1, 2019	Fifth draft.
v0.6	Dec. 2, 2019	Sixth draft.
v1.0	Dec. 16, 2019	Initial release of DAOS VOL connector.

Contents

The HDF Group

The HDF Group

List of Figures

The HDF Group

Acronyms

DAOS Distributed Asynchronous Object Storage. 5, 7–13, 20, 33–35, 39

HDF5 Hierarchical Data Format, v5. 5, 7–13, 20, 33–35

VOL Virtual Object Layer. 5, 7–13, 20, 33–35, 39

Glossary

connector A VOL connector is a thin layer between HDF5 and the storage. It translates HDF5 VOL storage related calls into actual storage operations.. 5, 7–13, 20, 33–35, 39
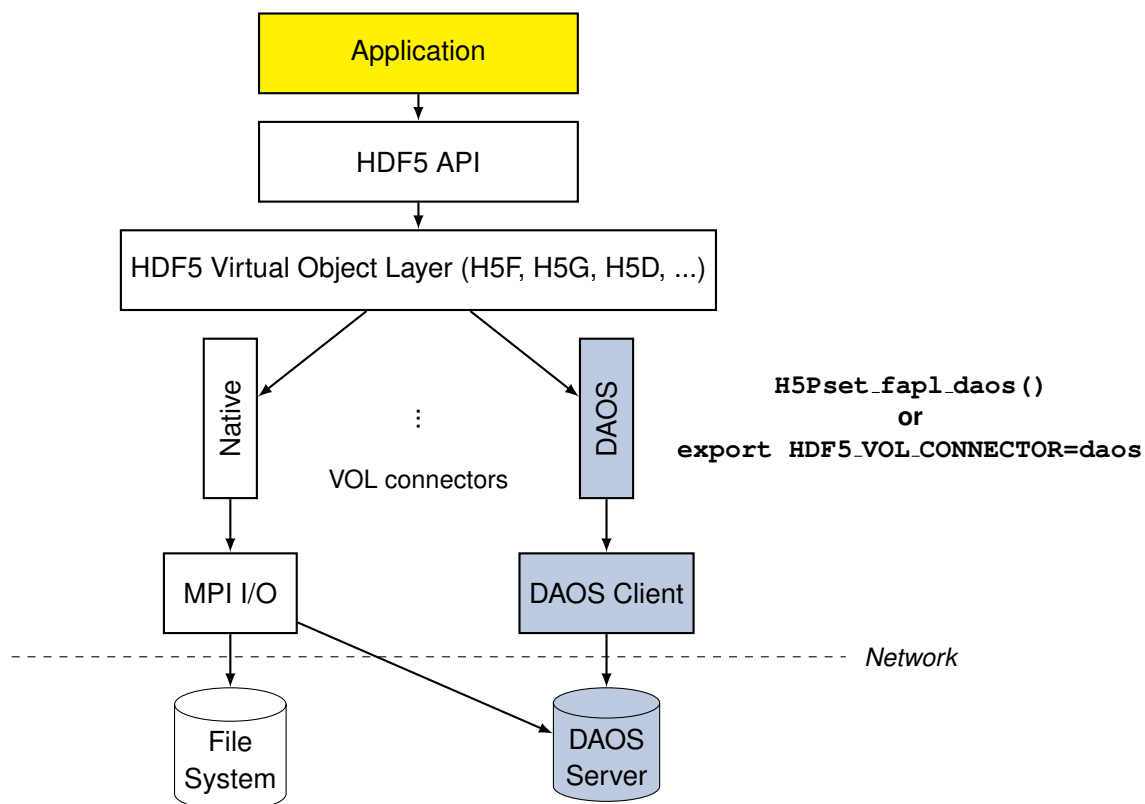
1. Overview

Figure 1 – DAOS within Virtual Object Layer. All of the HDF5 I/O related calls are routed to the DAOS VOL connector.

2. Using the DAOS VOL connector within an HDF5 application

This section outlines the unique aspects of writing, building and running HDF5 applications with the DAOS VOL connector.

2.1. Building the HDF5 DAOS VOL connector

The following is a quick set of instructions for building the DAOS VOL connector connector. Note that these instructions are not comprehensive and may be subject to change in future releases; please refer to the DAOS VOL connector's README file for the most up to date instructions.

The DAOS VOL connector is built using CMake. CMake version 2.8.12.2 or greater is required to build the connector itself, but version 3.1 or greater is required to build the connector's tests. To build the connector, one should create a build directory within the source tree:

```
cd daos-vol
mkdir build
cd build
```

After that, if all of the required components (DAOS, CaRT, MPI and HDF5) are located within the system path, building the connector should be as simple as running the following two commands to have CMake generate the build files for `make` to use.

```
ccmake ..
make && make install
```

Some notable CMake variables are listed below. These can be used to control the build process and can be supplied to the `cmake` command by prepending them with `-D` or turned on in `ccmake`. Some of the connector-specific options may be needed if the required components mentioned previously cannot be found within the system path.

CMake-specific options:

- `CMAKE_INSTALL_PREFIX` — This variable controls the install directory that the resulting output files are written to.

- `CMAKE_BUILD_TYPE` — This variable controls the type of build used for the VOL connector. Valid values are Release, Debug, RelWithDebInfo and MinSizeRel. (*Default: RelWithDebInfo*)

DAOS VOL connector-specific options:

- `BUILD_TESTING` — This variable is used to enable/disable building of the DAOS VOL connector's tests.

- `BUILD_EXAMPLES` — This variable is used to enable/disable building of the DAOS VOL connector's HDF5 examples.

- `CART_INCLUDE_DIR` — This variable controls the CaRT include directory used by the VOL connector build process. Used in conjunction with the `CART_LIBRARY` variable.

The HDF Group

- `CART_LIBRARY` — This variable controls the CaRT library used by the VOL connector build process. It should be set to the full path to the CaRT library, including the library's name (e.g., /path/libcart.so). Used in conjunction with the `CART_INCLUDE_DIR` variable.

- `DAOS_LIBRARY` — This variable controls the DAOS library used by the VOL connector build process. It should be set to the full path to the DAOS library, including the library's name (e.g., /path/libdaos.so). Used in conjunction with the `DAOS_COMMON_LIBRARY` and `DAOS_INCLUDE_DIR` variables.

- `DAOS_COMMON_LIBRARY` — This variable controls the DAOS 'common' library used by the VOL connector build process. It should be set to the full path to the DAOS common library, including the library's name (e.g., /path/libdaos_common.so). Used in conjunction with the `DAOS_LIBRARY` and `DAOS_INCLUDE_DIR` variables.

- `DAOS_INCLUDE_DIR` — This variable controls the DAOS include directory used by the VOL connector build process. Used in conjunction with the `DAOS_LIBRARY` and `DAOS_COMMON_LIBRARY` variables.

- `MPI_C_COMPILER` — This variable controls the MPI C Compiler used by the VOL connector build process. It should be set to the full path to the MPI C Compiler, including the name of the executable.

- `HDF5_C_COMPILER_EXECUTABLE` — This variable controls the HDF5 compiler wrapper script used by the VOL connector build process. It should be set to the full path to the HDF5 compiler wrapper, including the name of the wrapper script. The following two variables may also need to be set.

- `HDF5_C_LIBRARY_hdf5` — This variable controls the HDF5 library used by the VOL connector build process. It should be set to the full path to the HDF5 library, including the library's name (e.g., /path/libhdf5.so). Used in conjunction with the `HDF5_C_INCLUDE_DIR` variable.

- `HDF5_C_INCLUDE_DIR` — This variable controls the HDF5 include directory used by the VOL connector build process. Used in conjunction with the `HDF5_C_LIBRARY_hdf5` variable.

2.2. Writing HDF5 DAOS VOL connector applications

There are currently two main ways to tell an existing HDF5 application to use the DAOS VOL connector: either *implicitly* by using environment variables to tell the HDF5 library to load the connector as a dynamically loaded plugin or *explicitly* by making use of HDF5 property lists.

2.2.1. With the DAOS VOL connector as a dynamically-loaded plugin

HDF5 has the capability to dynamically load and use a VOL connector for running applications with. In order to choose a particular VOL connector to use, two initial steps must be taken. First, one must help HDF5 locate the VOL connector by pointing to the directory which contains the built library. This can be accomplished by setting the environment variable `HDF5_PLUGIN_PATH` to this directory. Next, HDF5 needs to know the name of which library to use, which is configured by setting the environment variable `HDF5_VOL_CONNECTOR` to the name of the connector.

In order to use the DAOS VOL connector, the aforementioned environment variables should be set as:

```
HDF5_PLUGIN_PATH=/daos/vol/installation/directory/lib
HDF5_VOL_CONNECTOR=daos
```

The HDF Group

Having completed this step, HDF5 will be setup to load the DAOS VOL connector and use it for running applications, including HDF5's own tests. No additional modifications will need to be made to the existing HDF5 application.

2.2.2. Without the DAOS VOL connector as a dynamically-loaded plugin

If dynamic loading of the DAOS VOL connector is not used, any HDF5 application using the connector must:

1. Include `daos_vol_public.h`, found in the `include` directory of the DAOS VOL connector installation directory.

2. Link against `libhdf5_vol_daos.so` (or similar), found in the `lib` directory of the DAOS VOL connector installation directory, and against `libuuid.so` (or similar) in order to use UUIDs. Note that dependencies can alternatively be retrieved through CMake or pkg-config.

An HDF5 DAOS VOL connector application also requires in that particular case three new function calls in addition to those for an equivalent HDF5 application (see Appendix A for more details):

■ `H5daos_init()` — Initializes the DAOS VOL connector

Called upon application startup, before any file is accessed.

■ `H5Pset_fapl_daos()` — Sets DAOS VOL connector access on File Access Property List.

Called to prepare a FAPL to open a file through the DAOS VOL connector. See HDF5 File Access Property Lists for more information about File Access Property Lists.

■ `H5daos_term()` — Cleanly shutdowns the DAOS VOL connector

Called on application shutdown, after all files have been closed.

2.2.3. Skeleton Example

Below is a no-op application that opens and closes a file using the DAOS VOL connector. For clarity, no error-checking is performed. Note that this example is meant only for the case when the DAOS VOL connector is not being dynamically loaded.

```c
#include "hdf5.h"
#include "daos_vol_public.h"

int main(void)
{
    uuid_t pool_uuid;
    hid_t fapl_id, file_id;

    /* Parse the pool UUID. */
    uuid_parse("fce30f79-b34b-46c1-9b1f-bb52d99dacca", pool_uuid);

    /* Initialize DAOS VOL connector using MPI_COMM_WORLD for the pool
     * communicator, the above parsed UUID for the pool UUID, "daos_server"
     * as the group name for the DAOS servers managing the pool and
     * simply rank 0 as the only rank in the pool service list. */
    H5daos_init(MPI_COMM_WORLD, pool_uuid, "daos_server", "0");
```

```
    fapl_id = H5Pcreate(H5P_FILE_ACCESS);
    H5Pset_fapl_daos(fapl_id, MPI_COMM_WORLD, MPI_INFO_NULL);

    /* Currently required for the DAOS VOL connector, set all metadata
     * operations to be collective */
    H5Pset_all_coll_metadata_ops(fapl_id, true);

    file_id = H5Fopen("my_file.h5", H5F_ACC_RDWR, fapl_id);

    /* Operate on file */
    [...]

    H5Pclose(fapl_id);
    H5Fclose(file_id);

    /* Terminate the DAOS VOL connector. */
    H5daos_term();

    return 0;
}
```

2.3. Building HDF5 DAOS VOL connector applications

Assuming an HDF5 application has been written following the instructions in the previous section, the application should be built as normal for any other HDF5 application. However, if the DAOS VOL connector is not being dynamically loaded, the steps in the following section are required to build the application.

2.3.1. Without the DAOS VOL connector as a dynamically-loaded plugin

To link in the required libraries, the compiler will likely require the additional linker flags:

```
-lhdf5_vol_daos -luuid
```

However, these flags may vary depending on platform, compiler and installation location of the DAOS VOL connector. It is highly recommended that compilation of HDF5 DAOS VOL connector applications be done using either the `h5cc/h5pcc` script included with HDF5 distributions, or CMake, pkg-config, as these will manage linking with the HDF5 library.

If HDF5 was built using autotools, this script will be called `h5pcc` and may be found in the `bin` directory of the HDF5 installation. If HDF5 was built with CMake, this script will simply be called `h5cc` and can be found in the same location. The above notice about additional library linking applies to usage of `h5cc/h5pcc`. For example:

```
h5cc/h5pcc -lhdf5_vol_daos -luuid my_application.c -o my_application
```

2.4. Running HDF5 DAOS VOL connector applications

Running applications that use the DAOS VOL connector connector requires access to a DAOS server. Refer to DAOS Software Installation for more information on the setup process for this. For the DAOS VOL connector to correctly interact with a DAOS server instance, the server must be running and it must be passed the UUID of the DAOS pool to use and a list of DAOS pool service list ranks, as detailed in the following sections.

2.4.1. Starting the DAOS Server

Instructions for starting a DAOS Server can be found in the DAOS Documentation.

2.4.2. With the DAOS VOL connector as a dynamically-loaded plugin

If the DAOS VOL connector is dynamically loaded by HDF5, the DAOS pool UUID and DAOS pool service rank list are passed via the two environment variables below.

```
DAOS_POOL - The UUID of the DAOS pool to use.

DAOS_SVCL - A comma-separated list of server ranks used for
            daos_pool_connect(). Generated from daos_pool_create().
```

2.4.3. Without the connector as a dynamically-loaded plugin

If the DAOS VOL connector is not being dynamically loaded, the DAOS pool UUID and DAOS pool service rank list should be passed via the call to H5daos_init() within the application.

2.4.4. Example Applications

Some of the example C applications which are included with HDF5 distributions have been adapted to work with the DAOS VOL connector and are included under the top-level examples directory in the DAOS VOL connector source root directory. The built example applications can be run from the bin directory inside the build directory.

In addition to these examples, the test/vol directory contains several test files, each containing test functions that are examples of HDF5 applications in miniature, focused on a particular behavior. These mini-application tests cover a moderate amount of HDF5's public API functionality and should be a good indicator of whether the DAOS VOL connector is working correctly in conjunction with a running DAOS API-aware instance. Note that these tests currently rely on HDF5's dynamically-loaded VOL connector capabilities in order to run with the DAOS VOL connector.

The HDF Group

3. HDF5 API Support

3.1. Feature Specific Support

The following sections serve to illustrate the DAOS VOL connector's support for features in HDF5, as well as to highlight any differences between the expected behavior of an HDF5 feature versus the actual behavior as implemented by the VOL connector.

3.1.1. Attribute Features

Feature			Supported?	Notes
Dataspace	Dimensionality	H5S_NULL	Yes	
		H5S_SCALAR	Yes	
		SIMPLE	Yes	
Datatype		Atomic	Yes	
		Compound	Yes	
		Variable-length	Yes	
		Array	Yes	
		Opaque	Yes	
		Reference	Yes	
Properties	Name Encoding	ASCII	Yes	
		UTF-8	No	

3.1.2. Dataset Features

Feature			Supported?	Notes
Dataspace	Dimensionality	H5S_NULL	Yes	
		H5S_SCALAR	Yes	
		SIMPLE	Yes	
	Selection Type	NONE	Yes	
		H5S_ALL	Yes	
		Hyperslab Selection	Yes	
		Point Selection	Yes	
Datatype		Atomic	Yes	
		Compound	Yes	
		Variable-length	Yes	
		Array	Yes	
		Opaque	Yes	
		Reference	Yes	

Feature			Supported?	Notes
Properties	Storage Properties (creation)	Compact	No	Setting is ignored; stored as contiguous.
		External	No	Setting is ignored; stored as contiguous.
		Contiguous	Yes	Default storage type.
		Chunked	Yes	
		VDS	No	The VDS feature is not currently planned to be supported.
	Other Properties (creation)	Attribute Creation Order	Yes	In order to work correctly, the attribute creation order feature requires that the dataset is touched collectively or that the application otherwise ensures no concurrent access to the dataset[1].
		Fill Value	Yes	
		Filters	No	HDF5 does not expose any public APIs for working with the filter pipeline; however, this feature may be supported in the future.
		Storage Allocation Time	N/A	

---

[1]This restriction may be removed in the future.

Feature			Supported?	Notes
Properties (cont.)	Access Properties	Chunk cache	No	HDF5 does not expose any public APIs for implementing a chunk cache for arbitrary VOL connectors; however, this feature may be supported in the future.
		VDS views and printf	No	The VDS feature is not currently planned to be supported.
		MPI-I/O Collective Metadata Ops	Yes	By default, all metadata operations are collective for writes and independent for reads.[1]
	Transfer Properties	MPI-I/O Independent or Collective I/O mode	N/A	

---

[1]Independent metadata writes will be supported in the future.

3.1.3. File Features

Feature		Supported?	Notes
File creation flags	H5F_ACC_TRUNC	Yes	The file creation flags behave as for native HDF5.
	H5F_ACC_EXCL	Yes	
File opening flags	H5F_ACC_RDWR	Yes	
	H5F_ACC_RDONLY	Yes	
Properties	Creation Properties	Attribute Creation Order	Yes	In order to work correctly, the attribute creation order feature requires that the dataset is touched collectively or that the application otherwise ensures no concurrent access to the dataset[1]. The rest of the file creation properties are related to the native HDF5-specific file format.

	Access Properties (Drivers)	SEC2 Driver	N/A	These drivers are applicable to native HDF5 only.
		Family Driver	N/A	
		Split Driver	N/A	
		Multi Driver	N/A	
		Core Driver	N/A	
		Log Driver	N/A	
		MPI-I/O	Yes	This property just indicates parallel access to the file; it doesn't use HDF5 MPI I/O driver underneath.

---

[1]This restriction may be removed in the future.

The HDF Group

Feature			Supported?	Notes
Properties (cont.)	Access Properties (Other)	MPI-I/O Collective Metadata Ops	Yes	By default, all metadata operations are collective for writes and independent for reads.[1]
		User block	N/A	
		Chunk Cache	No	HDF5 does not expose any public APIs for implementing a chunk cache for arbitrary VOL connectors; however, this feature may be supported in the future.
		Object flushing callbacks	N/A	
		File closing degree	N/A	
		Evict on close	N/A	
		Sieve buffer size for partial I/O	No	HDF5 does not expose any public APIs for implementing a partial I/O sieve buffer for arbitrary VOL connectors; however, this feature may be supported in the future.
		File Image	N/A	

---

[1]Independent metadata writes will be supported in the future.

3.1.4. Group Features

Feature			Supported?	Notes
Properties	Creation Properties	Link Creation Order	Yes	In order to work correctly, the link creation order feature requires that the parent group is touched collectively or that the application otherwise ensures no concurrent access to the group.[2]
		Attribute Creation Order	Yes	In order to work correctly, the attribute creation order feature requires that the dataset is touched collectively or that the application otherwise ensures no concurrent access to the dataset.[2]
		Other Properties	N/A	These properties are related to the native HDF5-specific file format.
	Access Properties	MPI-I/O Collective Metadata Ops	Yes	By default, all metadata operations are collective for writes and independent for reads.[1]

---

[1]Independent metadata writes will be supported in the future.

[2]This restriction may be removed in the future.

3.2. API Specific Support

The following sections serve to illustrate the DAOS VOL connector's support for the HDF5 API, as well as to highlight any differences between the expected behavior of an HDF5 API call versus the actual behavior as implemented by the VOL connector. If a particular HDF5 API call does not appear among these tables, it is most likely a native HDF5-specific API call which cannot be implemented by non-native HDF5 VOL connectors. These types of API calls are listed among the tables in Appendix B.

3.2.1. H5A interface

Supported API calls

API call	Notes
H5Acreate(1/2)	
H5Acreate_by_name	
H5Aopen(_by_name/_by_idx)	For H5Aopen_by_idx, `H5_ITER_DEC` is currently unsupported for the index ordering when `H5_INDEX_NAME` is used for the index type
H5Aopen_idx	Deprecated in favor of H5A_open_by_idx
H5Aopen_name	Deprecated in favor of H5A_open_by_name
H5Awrite	
H5Aread	
H5Aclose	
H5Aiterate(2)	- Restarting iteration from an index value is currently unsupported[1] `H5_ITER_DEC` is currently unsupported for the index ordering when `H5_INDEX_NAME` is used for the index type
H5Aiterate_by_name	- Restarting iteration from an index value is currently unsupported[1] `H5_ITER_DEC` is currently unsupported for the index ordering when `H5_INDEX_NAME` is used for the index type
H5Aexists(_by_name)	
H5Arename(_by_name)	
H5Adelete(_by_name/_by_idx)	For H5Adelete_by_idx, `H5_ITER_DEC` is currently unsupported for the index ordering when `H5_INDEX_NAME` is used for the index type
H5Aget_name(_by_idx)	For H5Aget_name_by_idx, `H5_ITER_DEC` is currently unsupported for the index ordering when `H5_INDEX_NAME` is used for the index type

---

[1]Will be supported by end of Q4 2019.

The HDF Group

API call	Notes
H5Aget_space	
H5Aget_type	
H5Aget_info(_by_name/_by_idx)	Of the four fields in the `H5A_info_t` struct: ■ `corder_valid` is set to TRUE only if attribute creation order tracking is enabled for the object containing the attribute; it is set to FALSE otherwise ■ `corder` is set appropriately if attribute creation order tracking is enabled for the object containing the attribute; it is set to 0 otherwise ■ `cset` is currently always set to `H5T_CSET_ASCII` ■ `data_size` is set appropriately For H5Aget_info_by_idx, `H5_ITER_DEC` is currently unsupported for the index ordering when `H5_INDEX_NAME` is used for the index type
H5Aget_create_plist	

Currently unsupported API calls

API call	Notes
H5Aget_storage_size	`H5Aget_storage_size` is not currently planned to be supported.

3.2.2. H5D interface

Supported API calls

API call	Notes
H5Dcreate(1/2)	
H5Dcreate_anon	
H5Dopen(1/2)	
H5Dwrite	
H5Dread	
H5Dclose	
H5Dextend	Upon dataset shrinking, data is currently not cleared.[1]
H5Dset_extent	Upon dataset shrinking, data is currently not cleared.[1]
H5Dget_space	
H5Dget_type	
H5Dget_create_plist	
H5Dget_access_plist	
H5Dget_space_status	Space status is currently always set to `H5D_SPACE_STATUS_NOT_ALLOCATED`
H5Dflush	`H5Dflush` is currently implemented as a no-op.
H5Drefresh	

Currently unsupported API calls

API call	Notes
H5Dget_storage_size	`H5Dget_storage_size` is not currently planned to be supported.

---

[1]Will be supported in the future.

The HDF Group

3.2.3. H5F interface

Supported API calls

API call	Notes
H5Fcreate	
H5Fopen	
H5Freopen	
H5Fis_accessible	
H5Fget_create_plist	
H5Fget_access_plist	
H5Fget_intent	
H5Fget_name	
H5Fget_obj_count	
H5Fget_obj_ids	
H5Fdelete	
H5Fflush	`H5Fflush` is currently implemented as a no-op.
H5Fclose	

Currently unsupported API calls

API call	Notes
H5Fmount	`H5Fmount` is not currently planned to be supported.
H5Funmount	`H5Funmount` is not currently planned to be supported.

3.2.4. H5G interface

Supported API calls

API call	Notes
H5Gcreate(1/2)	
H5Gcreate_anon	
H5Gopen(1/2)	
H5Gclose	
H5Gunlink	
H5Gget_create_plist	
H5Gget_info(_by_name/_by_idx)	Of the four fields in the `H5G_info_t` struct: ■ `storage_type` is always set to `H5G_STORAGE_TYPE_UNKNOWN` ■ `nlinks` is set appropriately ■ `max_corder` is set appropriately if link creation order is tracked for the group ■ `mounted` is currently always set to `FALSE` For `H5Gget_info_by_idx`, `H5_ITER_DEC` is currently unsupported for the index ordering when `H5_INDEX_NAME` is used for the index type
H5Gget_linkval	
H5Gget_num_objs	
H5Gget_objname_by_idx	`H5_ITER_DEC` is currently unsupported for the index ordering when `H5_INDEX_NAME` is used for the index type
H5Glink(2)	Currently only hard and soft link creation are supported[1]
H5Gmove(2)	Refer to Notes for `H5Lmove`
H5Gflush	`H5Gflush` is currently implemented as a no-op.
H5Grefresh	`H5Grefresh` is currently implemented as a no-op.

Currently unsupported API calls

API call	Notes

---

[1]External links are not currently planned to be supported.

3.2.5. H5L interface

Supported API calls

API call	Notes
H5Lcreate_hard	Reference count tracking is not currently implemented, so objects will not be removed when the last hard link pointing to them is removed[1]
H5Lcreate_soft	
H5Lexists	
H5Literate(_by_name)	■ Restarting iteration from an index value is currently unsupported[1] ■ `H5_ITER_DEC` is currently unsupported for the iteration order when `H5_INDEX_NAME` is used for the index type
H5Lvisit(_by_name)	■ Restarting iteration from an index value is currently unsupported[1] ■ `H5_ITER_DEC` is currently unsupported for the iteration order when `H5_INDEX_NAME` is used for the index type
H5Ldelete	Reference count tracking is not currently implemented, so objects will not be removed when the last hard link pointing to them is removed[1]
H5Ldelete_by_idx	`H5_ITER_DEC` is currently unsupported for the index ordering when `H5_INDEX_NAME` is used for the index type

---

[1]Will be supported by end of Q4 2019.

The HDF Group

API call	Notes
H5Lget_info	Of the five fields in the `H5L_info_t` struct: ■ `type` is set appropriately ■ `corder_valid` is set to TRUE only if link creation order tracking is enabled for the group containing the link; it is set to FALSE otherwise ■ `corder` is set appropriately if link creation order tracking is enabled for the group containing the link; it is set to 0 otherwise ■ `cset` is currently always set to `H5T_CSET_ASCII` ■ `u` has member `address` or `val_size` set appropriately based on whether the link is a hard link or not
H5Lget_info_by_idx	`H5_ITER_DEC` is currently unsupported for the index ordering when `H5_INDEX_NAME` is used for the index type
H5Lget_val	
H5Lget_val_by_idx	`H5_ITER_DEC` is currently unsupported for the index ordering when `H5_INDEX_NAME` is used for the index type
H5Lget_name_by_idx	`H5_ITER_DEC` is currently unsupported for the index ordering when `H5_INDEX_NAME` is used for the index type

API call	Notes

H5Lcopy	Currently no support for the following properties:
	
	■ LAPL
	    – H5Pset_nlinks
	    – H5Pset_elink_prefix[1]

H5Lmove	Currently no support for the following properties:
	
	■ LAPL
	    – H5Pset_nlinks
	    – H5Pset_elink_prefix[1]

Currently unsupported API calls

API call	Notes
H5Lcreate_external	`H5Lcreate_external` is not currently planned to be supported. As DAOS containers can contain large amounts of objects, the necessity for external links is lessened as compared to a traditional storage system.
H5Lcreate_ud	`H5Lcreate_ud` is not currently planned to be supported.

---

[1]External links are not currently planned to be supported.

3.2.6. H5O interface

Supported API calls

API call	Notes
H5Oopen	
H5Oopen_by_addr	
H5Oopen_by_idx	`H5_ITER_DEC` is currently unsupported for the index ordering when `H5_INDEX_NAME` is used for the index type
H5Oclose	
H5Olink	
H5Oexists_by_name	
H5Ovisit(1/2)	`H5_ITER_DEC` is currently unsupported for the index ordering when `H5_INDEX_NAME` is used for the index type
H5Ovisit_by_name(1/2)	`H5_ITER_DEC` is currently unsupported for the index ordering when `H5_INDEX_NAME` is used for the index type

H5Ocopy	Currently no support for the following properties:
	
	■ OCpyPL
	
	   – `H5O_COPY_EXPAND_EXT_LINK_FLAG`[2]
	
	   – `H5O_COPY_EXPAND_REFERENCE_FLAG`[1]
	
	   – `H5O_COPY_MERGE_COMMITTED_DTYPE_FLAG`

H5Oflush	`H5Oflush` delegates to the appropriate `H5Xflush` routine based upon the given object's type, but that particular object's flush routine may be implemented as a no-op.
H5Orefresh	`H5Orefresh` delegates to the appropriate `H5Xrefresh` routine based upon the given object's type, but that particular object's refresh routine may be implemented as a no-op.

---

[1]Will be supported in the future.

[2]External links are not currently planned to be supported.

The HDF Group

Currently unsupported API calls

API call	Notes
H5Oincr_refcount[1]	
H5Odecr_refcount[1]	

---

[1]Will be supported by end of Q4 2019.

3.2.7. H5R interface

Supported API calls

API call	Notes
H5Rcreate_object	Object lookup and retrieval of a container's info (which are needed for `H5Rcreate_object`) are supported.
H5Rcreate_region	Object lookup and retrieval of a container's info (which are needed for `H5Rcreate_region`) are supported.
H5Rcreate_attr	Object lookup and retrieval of a container's info (which are needed for `H5Rcreate_attr`) are supported.
H5Ropen_object	Object opening (which is needed for `H5Ropen_object`) is supported.
H5Ropen_region	Object opening and retrieval of a dataset's dataspace (which are needed for `H5Ropen_region`) are supported.
H5Ropen_attr	Object opening and attribute opening (which are needed for `H5Ropen_attr`) are supported.
H5Rget_obj_type3	
H5Rget_file_name	Retrieval of a file's name (which is needed for `H5Rget_file_name`) is supported.

Currently unsupported API calls

API call	Notes
H5Rget_obj_name	

The HDF Group

3.2.8. H5T interface

Supported API calls

API call	Notes
H5Tcommit(1/2)	
H5Tcommit_anon	
H5Topen(1/2)	
H5Tclose	
H5Tget_create_plist	
H5Tflush	`H5Tflush` is currently implemented as a no-op.
H5Trefresh	`H5Trefresh` is currently implemented as a no-op.

Currently unsupported API calls

API call	Notes

3.3. Known Limitations

The following sections outline the known current limitations of the DAOS VOL connector.

3.3.1. Limitations in regards to the HDF5 API

■ If an application abnormally exits, the DAOS VOL connector currently leaves the file in an unusable state. Currently, the only way to re-use the same filename after an application interruption is to use a new DAOS pool. This issue will be resolved when rollback to a previous snapshot is supported.

3.3.2. Limitations in regards to DAOS

■ Following the previous point about application abnormal exits, as DAOS does not currently support forced container deletion, trying to overwrite an existing HDF5 file using the H5F_ACC_TRUNC flag when the file was left in an unusable state will fail; the error *"can't destroy container: generic I/O error (DER_IO)"* will be returned.

■ No support for conditional key insert/remove.

■ There is currently no support for distributed transactions.

The HDF Group

4. Testing the DAOS VOL connector

The following sections cover how to test the DAOS VOL connector, as well as the individual components of the DAOS VOL connector's overall testing infrastructure.

4.1. With CTest

Once the DAOS VOL connector has been built, running the connector's tests should be as simple as running

```
ctest .
```

from the build directory. This will run each of the VOL connector's test components in turn. For more information on using CTest's options to control testing behavior, refer to the CTest Documentation.

4.2. Manually

If testing the DAOS VOL connector without CTest, refer to Starting the DAOS Server and Running HDF5 DAOS VOL connector applications to make sure that the DAOS Server is up and running and your environment is setup correctly. Once that is done, the DAOS VOL connector's tests can be run directly from the `bin` directory inside the build directory. For a listing of the different test executables and the functionality they test, refer to the following sections.

4.3. DAOS VOL connector's testing components

4.3.1. Generic HDF5 VOL connector test suite

In order to test HDF5 VOL connectors to make sure that they are functioning as expected, a suite of tests which only use the public HDF5 API has been written. This suite of tests is available under the path:

`test/vol`

and when built, will appear as the `h5vl_test` and `h5vl_test_parallel` executables in the `bin` directory inside the build directory.

Note that running this test suite requires that your environment is setup to have HDF5 dynamically load the DAOS VOL connector. Also, this test suite currently does not have the capability to query what kind of functionality an HDF5 VOL connector supports and therefore certain tests will be skipped if they use an HDF5 API call which is not implemented, or which is specifically unsupported, by the DAOS VOL connector.

4.3.2. DAOS VOL connector-specific test suite

In addition to the generic VOL connector testing suite, the DAOS VOL connector also includes the following test suites, which test features specific to the connector:

- DAOS VOL connector Map test suite

This test suite tests the DAOS VOL connector against the 'Map' functionality in HDF5, which concerns map objects that store key-value pairs. When built, this test suite will appear as the `h5daos_test_map` and `h5daos_test_map_parallel` executables in the `bin` directory inside the build directory.

■ DAOS VOL connector Recovery testing suite

This test suite tests the DAOS VOL connector's ability to recover from a fault that causes the DAOS server or the HDF5 application to stop functioning. It also ensures the integrity of both metadata and raw data in a file after such a fault. When built, this test suite will appear as the `h5daos_test_recovery` executable in the `bin` directory inside the build directory.

A. Reference Manual

A.1. H5daos_init

Synopsis:

```
herr_t H5daos_init(MPI_Comm pool_comm, uuid_t pool_uuid, const char *pool_grp,
    const char *pool_svcl);
```

Purpose:

Initialize the DAOS VOL connector.

Description:

H5daos_init initializes the VOL connector by connecting to the pool and registering the connector with the library. pool_comm identifies the communicator used to connect to the DAOS pool. This should include all processes that will participate in I/O. This call is collective across pool_comm.

Parameters:

MPI_Comm pool_comm	IN: Communicator used for connecting to the pool
uuid_t pool_uuid	IN: UUID to identify a pool
const char *pool_grp	IN: Process set name of the DAOS servers managing the pool
const char *pool_svcl	IN: Comma-separated list of pool service replica ranks

Returns:

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

A.2. H5daos_term

Synopsis:

herr_t H5daos_term(**void**);

Purpose:

Terminate the DAOS VOL connector.

Description:

H5daos_term terminates the DAOS VOL connector.

Parameters:

None.

Returns:

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

A.3. H5Pset_fapl_daos

Synopsis:

```
herr_t H5Pset_fapl_daos(hid_t fapl_id, MPI_Comm comm, MPI_Info info);
```

Purpose:

Set the file access property list to use the DAOS VOL connector.

Description:

H5Pset_fapl_daos modifies the file access property list to use the DAOS VOL connector. file_comm and file_info identify the communicator and info object used to coordinate actions on file create, open, flush, and close.

Parameters:

hid_t fapl_id	IN: File access property list ID
MPI_Comm file_comm	IN: MPI Communicator
MPI_Info file_info	IN: MPI Info

Returns:

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

B. Native HDF5 VOL connector-specific API calls

The following HDF5 API calls are either specific to the native HDF5 VOL connector or are not routed through the VOL and thus are not able to be implemented by the DAOS VOL connector (or other VOL connectors):

B.1. H5A interface

API call	Notes
H5Aiterate1	Deprecated in favor of H5Aiterate2
H5Aget_num_attrs	Deprecated in favor of H5Oget_info

B.2. H5D interface

API call	Notes
H5Dformat_convert	
H5Dget_offset	
H5Dget_chunk_index_type	
H5Dget_chunk_storage_size	
H5Dvlen_reclaim	
H5Dvlen_get_buf_size	
H5Diterate	
H5Dscatter	
H5Dgather	
H5Dfill	
H5Dread_chunk	
H5Dwrite_chunk	

B.3. H5F interface

API call	Notes
H5Fis_hdf5	Uses a default FAPL so can only ever be routed through the native HDF5 VOL connector
H5Fget_vfd_handle	
H5Fget_freespace	
H5Fget_filesize	
H5Fget_file_image	
H5Fget_mdc_config	
H5Fset_mdc_config	
H5Fget_mdc_hit_rate	
H5Fget_mdc_size	
H5Freset_mdc_hit_rate_stats	
H5Fget_info(1/2)	
H5Fget_metadata_read_retry_info	
H5Fget_free_sections	
H5Fclear_elink_file_cache	
H5Fstart_swmr_write	
H5Fstart_mdc_logging	
H5Fstop_mdc_logging	
H5Fget_mdc_logging_status	
H5Fset_libver_bounds	
H5Fformat_convert	
H5Freset_page_buffering_stats	
H5Fget_page_buffering_stats	
H5Fget_mdc_image_info	
H5Fget_eoa	
H5Fincrement_filesize	
H5Fget_dset_no_attrs_hint	
H5Fset_dset_no_attrs_hint	
H5Fset_latest_format	
H5Fset_mpi_atomicity	
H5Fget_mpi_atomicity	

B.4. H5G interface

API call	Notes
H5Gset_comment	Deprecated in favor of H5Oset_comment/H5Oset_comment_by_name
H5Gget_comment	Deprecated in favor of H5Oget_comment/H5Oget_comment_by_name
H5Giterate	Deprecated in favor of H5Literate
H5Gget_objinfo	Deprecated in favor of H5Lget_info/H5Oget_info
H5Gget_objtype_by_idx	Deprecated in favor of H5Lget_info/H5Oget_info

B.5. H5L interface

API call	Notes
H5Lregister	
H5Lunregister	
H5Lis_registered	
H5Lunpack_elink_val	

B.6. H5O interface

API call	Notes
H5Oget_info(1/2)	
H5Oget_info_by_name(1/2)	
H5Oget_info_by_idx(1/2)	
H5Oset_comment(_by_name)	Deprecated in favor of using attributes on objects
H5Oget_comment(_by_name)	Deprecated in favor of using attributes on objects
H5Oare_mdc_flushes_disabled	
H5Oenable_mdc_flushes	
H5Odisable_mdc_flushes	

The HDF Group

B.7. H5R interface

API call	Notes
H5Rdestroy	
H5Rget_type	
H5Requal	
H5Rcopy	
H5Rget_attr_name	

B.8. H5T interface

API call	Notes