RFC: Virtual Object Layer (VOL) API Compatibility

Dana Robinson

Not all VOL connectors will implement the full HDF5 API so application and connector authors need to be able to specify implementation requirements and feature sets, respectively.

This document introduces some proposed feature sets which can be implemented via VOL API compatibility flags. Their potential role in testing VOL connectors using the existing HDF5 test harness (with some modifications) is also discussed and maps of proposed flags to HDF5 API calls and VOL callbacks are provided.

This RFC assumes familiarity with the VOL.

Introduction

The HDF5 virtual object layer (VOL) provides a mechanism for mapping the HDF5 API to arbitrary storage. After deciding how HDF5 files, objects, and metadata will be stored via the new storage mechanism, a VOL connector plugin can be written which implements this new storage scheme in place of the native HDF5 file format. Using the new VOL connector is largely transparent, requiring only minimal setup in the application to configure and load the VOL connector prior to creating HDF5 storage.

A problem for both authors of VOL connectors and the application authors which will use them is negotiating the API contract between them. Most applications use only a small fraction of the HDF5 API and some HDF5 API calls may be difficult to implement in a particular storage medium. In light of the potential for mismatch, it would be very useful to have a mechanism that would allow an application to query a VOL connector concerning its capabilities before making HDF5 API calls.

Proposed API Compatibility Groups

Some proposed API compatibility groups are listed here. These groups will probably be split into smaller sets to aid development. For example the BASIC flags might be broken down into BASIC_FILE, BASIC_GROUP, etc. to allow developers to test connectors as they implement the the VOL callbacks for an HDF5 file object.

Note that not all HDF5 API calls go through the VOL. Dataspace (H5S) calls, for example, are VOL-independent and the capability flags will not apply to them.

A companion spreadsheet that more clearly lays out the groupings is located at https://bitbucket.hdfgroup.org/projects/HDFFV/repos/hdf5doc/browse/RFCs/HDF5/VOL/VOL_API_Calls_by_Feature_Flag_v1.xlsx

H5VL_API_CMP_FLAG_BASIC/ADVANCED/FULL

The idea of the these compatibility groups is to cover the most commonly used and easily implemented API calls. BASIC would cover the API calls covered by the most important callbacks (usually create/open/close). The calls in the get and specific callbacks would mostly be split between the ADVANCED and FULL flags, the latter being reserved for less commonly used or more difficult to implement functionality.

As noted earlier, it is likely that these calls will be further broken down based on (roughly) the callback groupings in the VOL. i.e.; file, group, dataset, datatype, attribute, link, object, reference, and possibly map. This would give you flags like H5VL_API_CMP_FLAG_DATASET_BASIC and allow more fine-grained descriptions of VOL connector functionality.

H5VL_API_CMP_FLAG_BY_IDX

The HDF5 API includes several functions that allow accessing a thing "by index", which could be either by name or by creation order. The latter may not be simple to implement in storage systems that do not track creation order, so this should have its own capability flag.

H5VL_API_CMP_FLAG_OBJECT_COPY

H5Ocopy() is a complicated API call in the native HDF5 file format and may be similarly difficult to implement in other storage systems.

H5VL_API_CMP_FLAG_MOUNT

H5Fmount() and H5Funmount() are not commonly used and may be difficult to implement correctly.

H5VL_API_CMP_FLAG_FLUSH_REFRESH

This flag may be needed to fully support storage systems with lazy persistence, where a write-flush-read cycle could return stale data.

H5VL_API_CMP_FLAG_NATIVE

This flag indicates that the VOL connector implements the API calls that require an equivalent of the native file format. This is most of use in the native VOL connector, but could be of use if someone were to write a terminal VOL connector that somehow mimicked the full functionality of the HDF5 in a way that were indistinguishable from the native VOL connector.

Implementation

HDF5 Library

An API compatibility flags field will be added to the existing H5VL_class_t struct used to define a VOL connector's info and callbacks. The struct already contains a cap_flags field, which is currently only used to indicate that a VOL connector is thread-safe, but it seems best to separate the API compatibility and other capabilities. An API call will also be added to the HDF5 library so that these flags can be obtained from a loaded VOL connector plugin.

The flags will be added as bitwise #defines in H5VLpublic.h.

VOL Connectors

VOL connector authors will simply need to decide on which capability flags they support and set this in the H5VL_class_t struct for the connector.

Use By Applications

Applications will use the query function to ensure that the features they require are implemented in the connector. It is up to the application to selectively enable and disable application features based on the presence or absence of flags or to reject non-conforming VOL connectors entirely. Calling unsupported functions will return error values from the HDF5 library but will not cause applications to crash.

The HDF5 command-line tools and HDFView will eventually be updated to query a VOL connector's capability flags, however a detailed plan for that is beyond the scope of this document.

Potential Use In A Future VOL-Aware HDF5 Test Harness

Thoroughly testing new VOL connectors is an ongoing concern. The existing HDF5 test suite that is built with the library has not yet been modified to handle arbitrary VOL connectors, so most VOL connector authors write their own test suite to ensure the VOL connector works as planned. Unfortunately, this is both extra work for the VOL connector author and almost certainly results in a test suite that is less comprehensive than that used by the HDF5 library and its two decades of experience probing the library's corner cases.

Although outside the scope of this document, The HDF Group is developing a plan to break the HDF5 library's tests up into groups based on the API compatibility flags described here. Combined with a mechanism for loading and configuring arbitrary VOL connectors, this would allow a VOL connector author to easily run their plugin through the subset of the HDF5 library's tests that support the connector's features.

Acknowledgements

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

Revision History

September 19, 2019: Version 1 circulated for comment within The HDF Group.

Appendix: HDF5 API Call Support By API Capability Flag

The companion spreadsheet at https://bitbucket.hdfgroup.org/projects/HDFFV/repos/hdf5doc/browse/RFCs/HDF5/VOL/VOL_API_Calls_by_Feature_Flag_v1.xlsx includes this.

Tables will be added to the final version of this document.

Appendix: VOL Callback Support By Feature Flag

This will be added to the companion spreadsheet listed above in the near future.

Tables will be added to the final version of this document.

Appendix: The H5VL_class_t Struct

```
/* Class information for each VOL connector */
typedef struct H5VL_class_t {
    /* Overall connector fields & callbacks */
    unsigned int version;                   /* VOL connector class struct version #     */
    H5VL_class_value_t value;               /* Value to identify connector              */
    const char *name;                       /* Connector name (MUST be unique!)         */
    unsigned cap_flags;                     /* Capability flags for connector           */
    herr_t (*initialize)(hid_t vipl_id);    /* Connector initialization callback        */
    herr_t (*terminate)(void);              /* Connector termination callback           */

    /* VOL framework */
    H5VL_info_class_t       info_cls;       /* VOL info fields & callbacks  */
    H5VL_wrap_class_t       wrap_cls;       /* VOL object wrap / retrieval callbacks */

    /* Data Model */
    H5VL_attr_class_t       attr_cls;       /* Attribute (H5A*) class callbacks */
    H5VL_dataset_class_t    dataset_cls;    /* Dataset (H5D*) class callbacks   */
    H5VL_datatype_class_t   datatype_cls;   /* Datatype (H5T*) class callbacks  */
    H5VL_file_class_t       file_cls;       /* File (H5F*) class callbacks      */
    H5VL_group_class_t      group_cls;      /* Group (H5G*) class callbacks     */
    H5VL_link_class_t       link_cls;       /* Link (H5L*) class callbacks      */
    H5VL_object_class_t     object_cls;     /* Object (H5O*) class callbacks    */

    /* Services */
    H5VL_request_class_t    request_cls;    /* Asynchronous request class callbacks */

    /* Catch-all */
    herr_t (*optional)(void *obj, hid_t dxpl_id, void **req, va_list arguments); /* Optional
callback */
```

```
} H5VL_class_t;
```

Appendix: H5VLget_api_capability_flags()

Name: `H5VLget_api_capability_flags`

Signature:

herr_t H5VLget_api_capability_flags(hid_t conn_id, /*out*/ unsigned int *flags)

Purpose:

Obtain the capability flags for a VOL connector.

Motivation:

Not all VOL connectors will implement the full HDF5 API. This query function allows applications to determine which functions a particular VOL connector supports. This allows an application to either reject a loaded VOL connector or engineer around the missing functionality.

Description:

This function queries a VOL connector for its feature flags, which allows applications to determine if a connector matches their HDF5 API requirements.

Parameters:

hid_t conn_id	An ID for a loaded and registered VOL connector.
unsigned int *flags	The feature flags supported by the VOL connector.

Returns:

A negative value on errors, non-negative on success.

Failure Modes:

This function will fail if conn_id is not an ID for a registered, loaded VOL connector or if flags is not a valid, non-NULL pointer.

Fortran Interface: `h5vlget_capability_flags_f`

History:

HDF5 1.12.0: Function added to the library (coincides with the VOL feature release)