HDF5 1.13.0
VFD and VOL Changes

September 24, 2021

The HDF Group

Dana Robinson
Joe Lee
The HDF Group

Outline

- HDF5 1.13.0
- VFD layer changes
- VOL changes
- Template VOL connectors
- Async and cache VOL discussion and demo

HDF5 1.13.0

HDF5 1.13.0 (unstable)

- Release date: Late 2021
  - Unstable release
    - APIs subject to change
    - File format may change
    - No binary compatibility guarantees
  - "Unstable" doesn't mean "buggy"
  - Please send us feedback if something could be improved!

- In the develop (default) branch in the hdf5 repository on GitHub
  - There will not be an hdf5_1_13 branch
  - All 1.13 releases will split off of the develop branch

- HDF5 1.14.0 (stable) targeted for late 2022

HDF5 1.13.0 (unstable)

- Assumes modern compilers and systems
  - C99
  - C++11 (if building C++ wrappers)
  - Targeted at current POSIX platforms
  - Visual Studio 2015+ (due to C99 needs)
  - Fewer checks for obsolete things than in 1.12/1.10

- Memory sanity checks OFF by default
  - Heap canaries cause problems with buffer (re)allocation/free

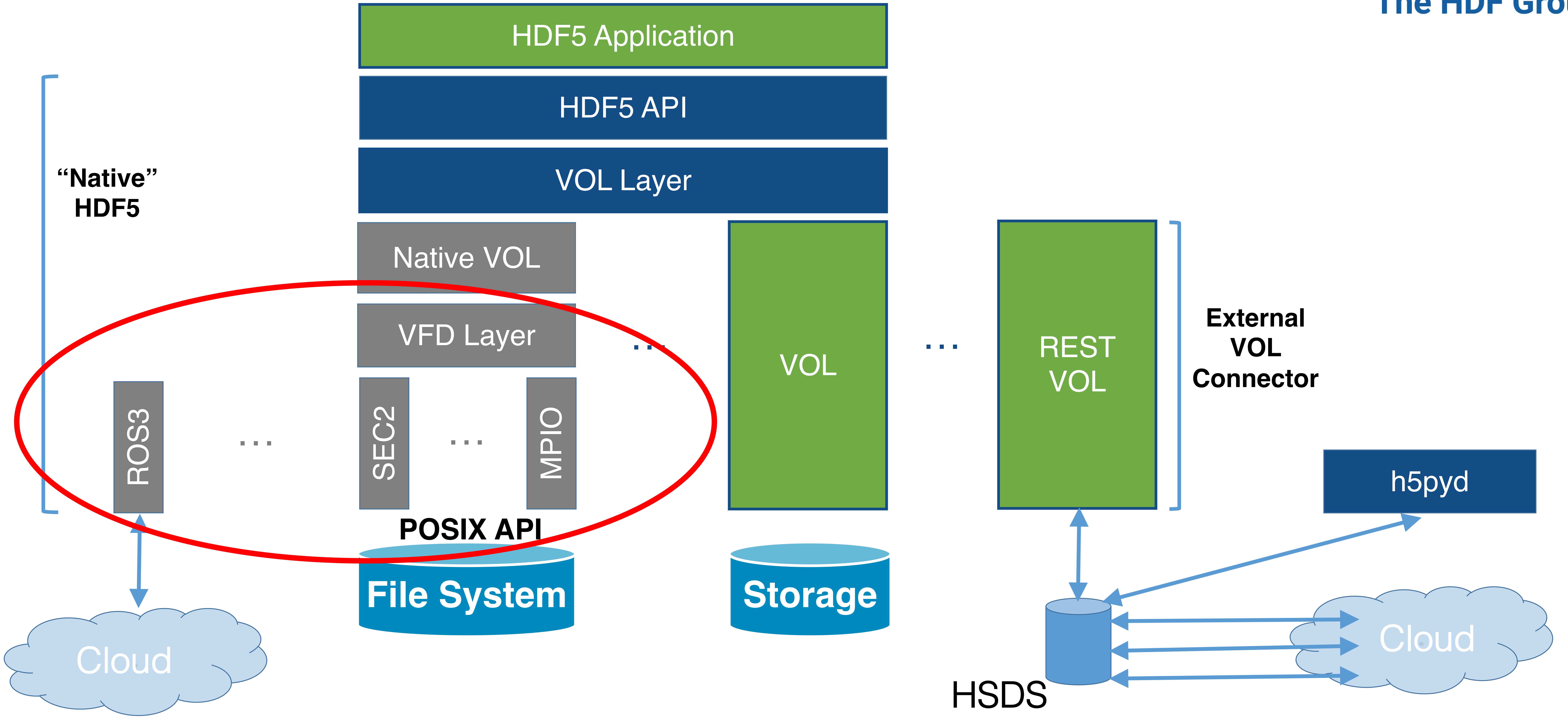
Other HDF5 1.13.0 Changes

- Asynchronous HDF5
  - New async versions of many API calls
  - New event set (H5ES) API calls
  - Requires an async-capable VOL connector (NOT in the native connector)

```
hid_t
H5Aopen(hid_t obj_id, const char *attr_name, hid_t aapl_id);

hid_t
H5Aopen_async(const char *app_file, const char *app_func,
              unsigned app_line, hid_t obj_id,
             const char *attr_name, hid_t aapl_id, hid_t es_id);
```

Virtual File Driver (VFD) Changes

HDF5 1.12 and Later Library Architecture



VFD Plugins - Motivation

- Connects the HDF5 native file format to a wider variety of data sources
- Leverages the strengths of the filter plugin scheme
- Convenience for users
- Welcomes new contributions to the HDF5 ecosystem
- More adaptable to a changing data landscape

Documentation

- VFD plugin RFC
  - https://github.com/HDFGroup/hdf5doc/blob/master/RFCs/HDF5_Library/VFL_DriverPlugins/RFC__A_Plugin_Interface_for_HDF5_Virtual_File_Drivers.pdf
- Support portal page for dynamic plugins of all types
  - https://portal.hdfgroup.org/display/HDF5/Dynamic+Plugins+in+HDF5
- Support portal page for registered plugins
  - https://portal.hdfgroup.org/display/support/Contributions
  - VFD page will be created after the feature is merged to develop

VFDs As Dynamically Loaded Plugins

- Works like existing filter and VOL plugins
- Same default paths (for all plugins)
  - POSIX: `/usr/local/hdf5/lib/plugin`
  - Windows: `%ALLUSERSPROFILE%/hdf5/lib/plugin`
- `HDF5_PLUGIN_PATH` environment variable
- VFDs are loaded when:
  - `H5Pset_driver*` is called
  - `HDF5_DRIVER` environment variable is set and a fapl is created

Environment Variables

- HDF5_DRIVER
    - Name of VFD to load
- HDF5_DRIVER_CONFIG
    - VFD configuration string
- Will replace the default VFD on the fapl at creation time

VFD Configuration Strings

- Some VFDs will require a way to pass arbitrary configuration data to them

- This can be done by passing a string that will be interpreted by the VFD
  - THG imposes NO structure on this string
  - Use whatever you like - JSON, YAML, XML, roll-your-own, etc.

- Used in `H5Pset*()` calls and HDF5 command-line tools (h5dump, etc.)

- VFD authors can call `H5Pget_driver_config_str()` on the passed-in fapl to get the string for processing

VFD 'values'

- New `H5FD_class_t` VFD struct field

- A unique identifier for the VFD plugin
  - NOT library-managed hid_t IDs (hence 'value' to avoid confusion)
  - Implemented as an integer (typedef'd to `H5FD_class_value_t`)

- Ranges reserved for specific purposes

• 0 - 255	for THG
• 256 - 511	for testing (will never be used by internal or external plugins)
• 512 - 65535	for new VFD plugins

- Register new VFD plugin values with THG (help@hdfgroup.org)

New API Calls

```
herr_t
H5Pset_driver_by_name(hid_t fapl_id, const char *driver_name,
                      const char *driver_config);

herr_t
H5Pset_driver_by_value(hid_t fapl_id, H5FD_class_value_t driver_value,
                      const char *driver_config);

ssize_t
H5Pget_driver_config_str(hid_t fapl_id, char *config_buf, size_t buf_size);
```

More New API Calls

```
htri_t
H5FDis_driver_registered_by_name(const char *driver_name);

htri_t
H5FDis_driver_registered_by_value(H5FD_class_value_t driver_value);
```

Tools Support

```
--vfd-value      Value (ID) of the VFD to use for
                 opening the HDF5 file specified

--vfd-name       Name of the VFD to use for opening
                 the HDF5 file specified

--vfd-info       VFD-specific configuration string
                 to pass to the VFD
```

- Same way you specify a VOL connector in 1.12

Authoring VFD Plugins

Need to implement `H5PLget_plugin_type()` and `H5PLget_plugin_info()`

The HDF5 library has two "example" VFDs:

- stdio - a single-file VFD
- multi - a multi-file VFD

Both use standard C and no internal HDF5 API calls so they can be copied to serve as templates for your own VFDs

No template project/repository at this time, but you could use the build files in the template VOL (link on a later slide) as a start.

ctl Callback

- New callback in `H5FD_class_t` struct
- Allows arbitrary operations by the VFD

ctl Callback

```
herr_t (*ctl)(H5FD_t *file, uint64_t op_code, uint64_t flags, const void *input,
                void **output);
```

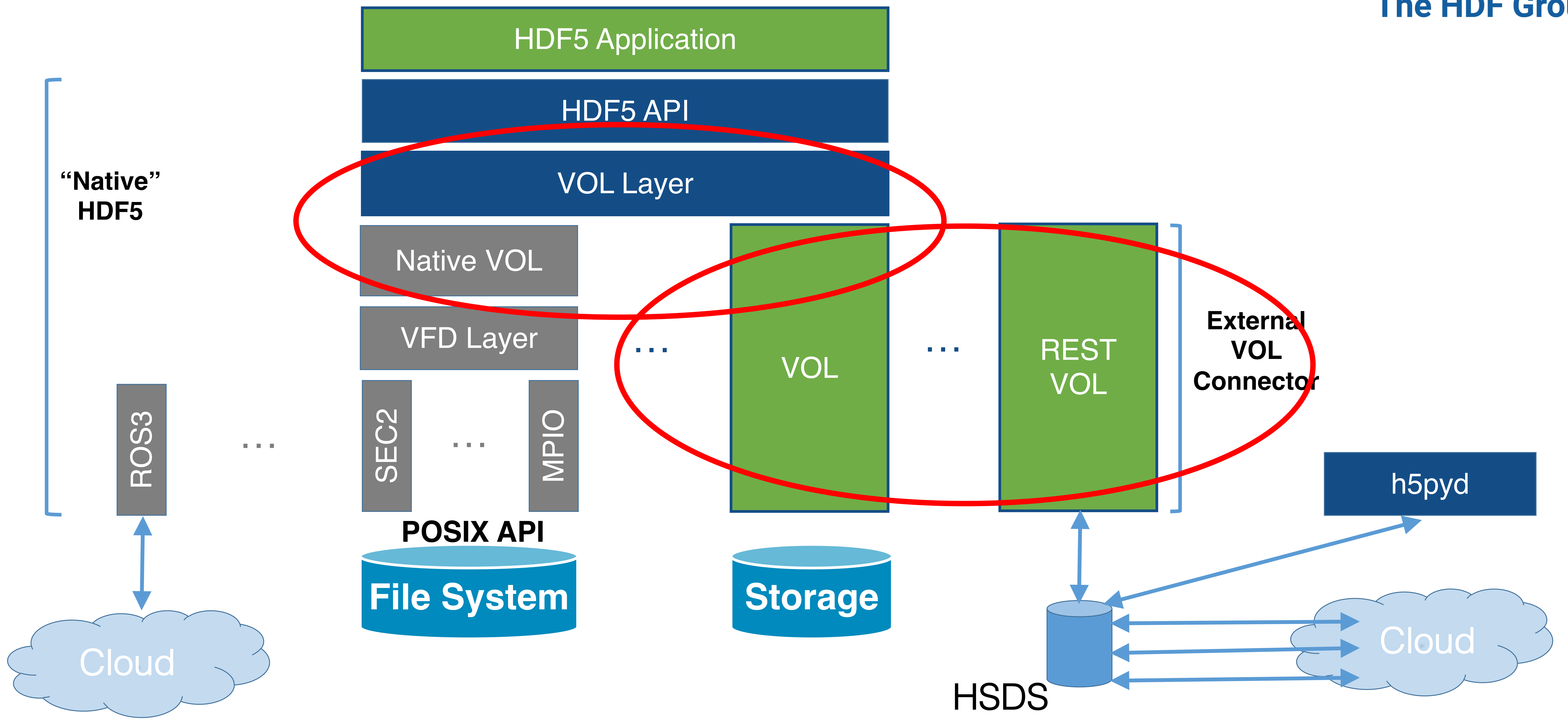
```
This newly-added "ctl" callback allows Virtual File Drivers to intercept and handle arbitrary
operations identified by an operation code. Its parameters are as follows:

`file` [in]    - A pointer to the file to be operated on
`op_code` [in] - The operation code identifying the operation to be performed
`flags` [in]   - Flags governing the behavior of the operation performed (see H5FDpublic.h
                 for a list of valid flags)
`input` [in]   - A pointer to arguments passed to the VFD performing the operation
`output` [out] - A pointer for the receiving VFD to use for output from the operation
```

VFD Testing

- The HDF5 test suite has been modified to separate library-VFD-specific VFD tests from VFD tests that should work for arbitrary drivers
- Simply set HDF5_DRIVER and HDF5_DRIVER_CONFIG and run 'make check' or CTest

Virtual Object Layer (VOL) Changes

HDF5 1.12 and Later Library Architecture



VOL Changes - Motivation

- The VOL is evolving as we gain experience with it
- New demands from ECP connectors
- Programming model is still fundamentally the same
- Biggest changes are for optional operations
- The changes that follow are mainly for connector authors

The HDF5 1.12 VOL API Is DEPRECATED

- Current VOL development should target 1.13.0
- Important changes to the VOL API cannot be brought to the 1.12 development branch due to binary compatibility issues
- Please switch your VOL development to target the develop branch!

Versioning

The HDF Group

- Current value of `H5VL_VERSION` is 2 (in `H5VLpublic.h`)
- Connectors now have their own version number, distinct from the VOL API version number

```
/* Overall connector fields & callbacks */
unsigned             version;            /**< VOL connector class struct version #     */
H5VL_class_value_t value;              /**< Value to identify connector              */
const char *         name;               /**< Connector name (MUST be unique!)         */
unsigned             conn_version;       /**< Version # of connector                   */
unsigned             cap_flags;          /**< Capability flags for connector           */
herr_t (*initialize)(hid_t vipl_id); /**< Connector initialization callback        */
herr_t (*terminate)(void);           /**< Connector termination callback           */
```

* NEW *

`specific/get` Callback Operation Changes

- Several of the `specific` and `get` callbacks have had changes to the operations they support

Added:

- `H5VL_ATTR_DELETE_BY_IDX`
- `H5VL_DATATYPE_GET_BINARY_SIZE`
- `H5VL_GROUP_MOUNT`
- `H5VL_GROUP_UNMOUNT`
- `H5VL_REQUEST_GET_ERR_STACK`
- `H5VL_REQUEST_GET_EXEC_TIME`

Removed:

- `H5VL_BLOB_GETSIZE`
- `H5VL_FILE_MOUNT`
- `H5VL_FILE_UNMOUNT`
- `H5VL_REQUEST_WAITALL`
- `H5VL_REQUEST_WAITANY`
- `H5VL_REQUEST_WAITSOME`

Callback Operation Changes

In other words:

- H5Adelete_by_idx is handled separately now
- Mount operations moved from file to group specific
- Blob "get size" specific callback moved to datatype get callback
- 1.12 does not have async, so the initial guesses at the request operations changed when the feature was implemented

No More Callback `va_list` Arguments

Old-style specific/get/optional callbacks

```
herr_t (*specific)(void *obj, H5VL_file_specific_t specific_type,
                   hid_t dxpl_id, void **req, va_list arguments);
```

New-style callbacks

```
herr_t (*specific)(void *obj, H5VL_file_specific_args_t *args,
                   hid_t dxpl_id, void **req);
```

Updated in both callbacks and associated "VOL developer" API calls like H5VLdataset_get()

No More Callback `va_list` Arguments

OLD

```
typedef enum H5VL_file_specific_t {
    H5VL_FILE_FLUSH,
    H5VL_FILE_REOPEN,
    H5VL_FILE_MOUNT,
    H5VL_FILE_UNMOUNT,
    H5VL_FILE_IS_ACCESSIBLE,
    H5VL_FILE_DELETE,
    H5VL_FILE_IS_EQUAL
} H5VL_file_specific_t;
```

```
/* Parameters for file 'specific' operations */
typedef struct H5VL_file_specific_args_t {
    H5VL_file_specific_t op_type;

    /* Parameters for each operation */
    union {
        /* H5VL_FILE_FLUSH */
        struct {
            H5I_type_t  obj_type;
            H5F_scope_t scope;
        } flush;

        /* <SNIP> */

        /* H5VL_FILE_IS_EQUAL */
        struct {
            void *   obj2;
            hbool_t *same_file;
        } is_equal;
    } args;
} H5VL_file_specific_args_t;
```

* NEW *

New `get_cap_flags` Callback

- Part of H5VL_introspect_class_t
- Capabilities flags are listed in `H5VLconnector.h`
- Added because stacked VOL connectors can't simply return their own capability flags
- Signature:

```
herr_t
(*get_cap_flags)(const void *info, unsigned *cap_flags);
```

H5VLquery_optional() Changes

The Boolean out parameter has been replaced with a set of bitwise flags to give a better sense of the operation's behavior

```
herr_t
H5VLquery_optional(hid_t obj_id, H5VL_subclass_t subcls, int opt_type,
                   hbool_t *supported);
```

Is now…

```
herr_t
H5VLquery_optional(hid_t obj_id, H5VL_subclass_t subcls, int opt_type,
                   uint64_t *flags);
```

H5VLquery_optional() Changes

```
/* Flags to return from H5VLquery_optional API and 'opt_query' callbacks */
H5VL_OPT_QUERY_SUPPORTED          0x0001 /* VOL connector supports this operation */
H5VL_OPT_QUERY_READ_DATA          0x0002 /* Operation reads data for object */
H5VL_OPT_QUERY_WRITE_DATA         0x0004 /* Operation writes data for object */
H5VL_OPT_QUERY_QUERY_METADATA     0x0008 /* Operation reads metadata for object */
H5VL_OPT_QUERY_MODIFY_METADATA    0x0010 /* Operation modifies metadata for object */
H5VL_OPT_QUERY_COLLECTIVE         0x0020
        /* Operation is collective (operations without this flag are assumed to be independent) */
H5VL_OPT_QUERY_NO_ASYNC           0x0040 /* Operation may NOT be executed asynchronously */
H5VL_OPT_QUERY_MULTI_OBJ          0x0080 /* Operation involves multiple objects */
```

Optional Operations

- Must be registered using `H5VLregister_opt_operation()`

```
herr_t
H5VLregister_opt_operation(H5VL_subclass_t subcls, const char *op_name,
                           int *op_val /*out*/)
```

- Library will assign an integer operation value that will be passed via `H5VL_optional_args_t` for checking in your `optional` callback.

- Avoids operation clash across VOL connectors

Optional Operations

The HDF Group

- This is more complicated now, and will be documented in further detail in the updated Connector Author's Guide
- Can look at the async or cache VOLs on GitHub for examples of how to handle optional operations
- Cache VOL connector: https://github.com/hpc-io/vol-cache
- Async VOL connector: https://github.com/hpc-io/vol-async

Async I/O

- The VOL has been updated for asynchronous operations
- Requires a suitable VOL (no native HDF5 async functionality)
- Presentation at HUG
  - Wednesday, October 13, 2021
  - https://www.hdfgroup.org/hug/hug21/agenda-hdf5-users-group-2021/

VOL Templates

Template and Passthru VOL Connectors

Template VOL Connector

- Template for constructing terminal VOL connectors
  - No functionality
  - Empty VOL structure
  - Autotools and CMake build files
  - You fill in the callbacks

- Located at:
  - https://github.com/HDFGroup/vol-template

Passthru VOL Connector

- Template for constructing pass-through VOL connectors
  - Simply forwards the calls to another VOL connector
  - You add anything sophisticated
  - Uses standard C and no internal HDF5 API calls so can be built externally

- Located in the HDF5 source repository
  - `src/H5VLpassthru.(c|h)`
  - Could copy into the template VOL connector to use its build files
  - Built and tested with the library (via 'make check-passthru-vol')

VOL Documentation

These are in the process of being updated (new versions in fall 2021)

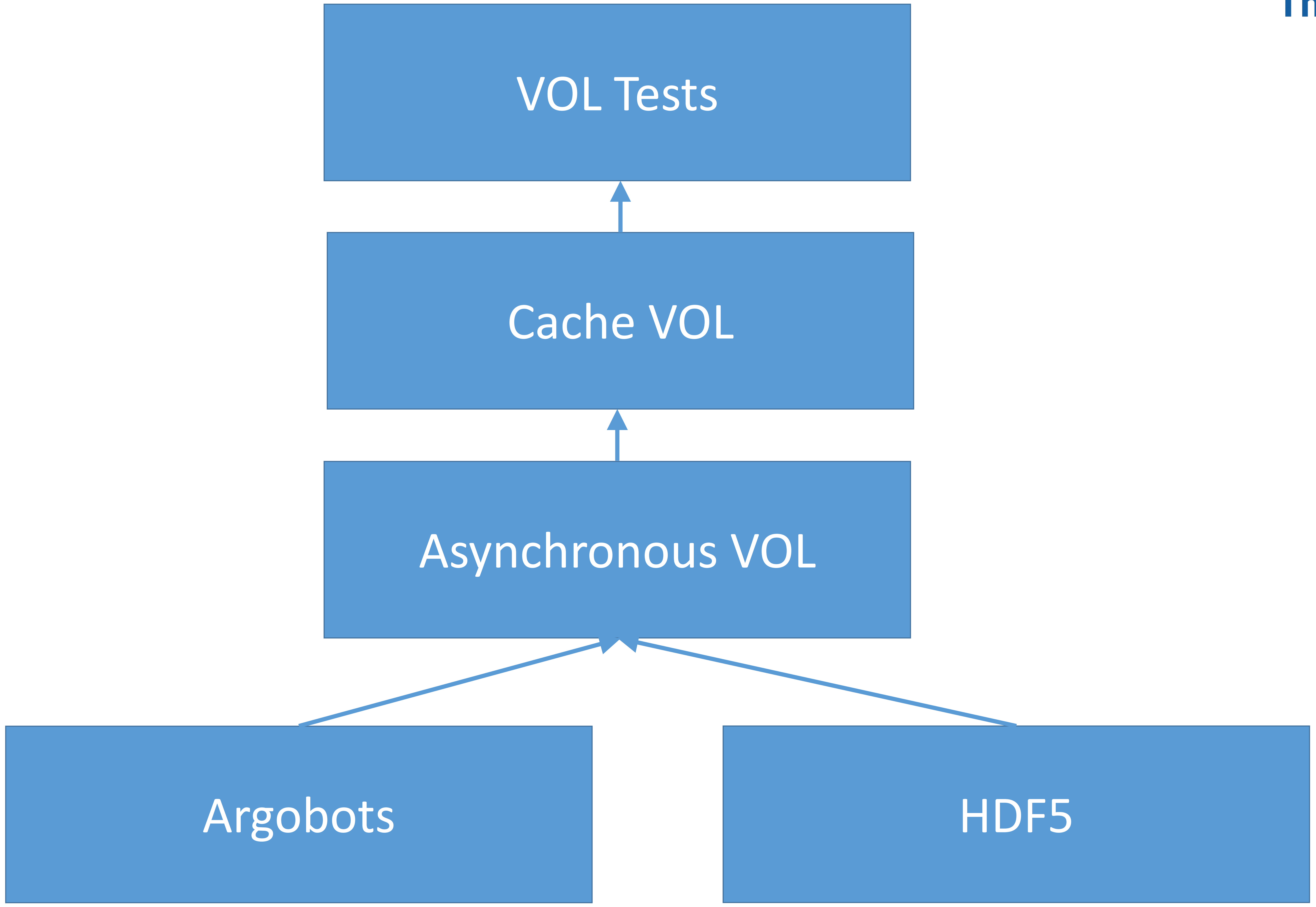
- VOL User's Guide
- VOL Connector Author Guide
- The updated CAG will have a section on migrating version 0 and 1 connectors
- Both can be found at https://portal.hdfgroup.org/display/HDF5/Virtual+Object+Layer

- VOL API Reference Manual
  - https://portal.hdfgroup.org/display/HDF5/Libraries+and+Tools+Reference
  - Moving to doxygen: https://docs.hdfgroup.org/hdf5/develop/

Asynchronous VOL and Cache VOL

Dependencies, Installation, and Testing

Dependencies



How to Install Async/Cache VOL using Spack

The HDF Group

3-liner Installation & Test Script:

1. $git clone https://github.com/hyoklee/spack
2. $cd spack/bin
3. $./test_vols.sh

HDF5 Users Group Meeting

The HDF Group

- October 12-15, 2021
- Async VOL discussion - October 13 @ 9:00 am
- Cache VOL discussion - October 13 @ 9:20 am

Demo

Live

Future Work

- Investigate **HDFGroup/vol-tests*** failures.
  - 3 tests fail out of 15.
  - 2 failures are different between **hpc-io/hdf5** and **HDFGroup/hdf5**.

- Push the development work (**hyoklee/spack**) to the official Spack repos.
  - **HDFGroup/hdf-spack**: The HDF Group reviews this repo.
  - **spack/spack**: Spack community reviews this repo.

*user/repo in GitHub

Acknowledgement

The HDF Group

This material is based upon work supported by the U.S. Department of Energy, Office of Science, under Contract Numbers DE-AC02-05CH11231 and DE-AC05-00OR22725.

THANK YOU!

Questions & Comments?

Register now to attend the HDF5 Users Group Meeting!

October 12-15

https://www.hdfgroup.org/hug/hug21/