Creating a Dataset

A dataset is a multidimensional array of data elements, together with supporting metadata. To create a dataset, the application program must specify the location at which to create the dataset, the dataset name, the datatype and dataspace of the data array, and the property lists.

Datatypes

A datatype is a collection of properties, all of which can be stored on disk, and which, when taken as a whole, provide complete information for data conversion to or from that datatype.

There are two categories of datatypes in HDF5:

- **Pre-defined:** These datatypes are opened and closed by HDF5.

  Pre-defined datatypes can be atomic or composite:
  - Atomic datatypes cannot be decomposed into smaller datatype units at the API level. For example: integer, float, reference, string.
  - Composite datatypes are aggregations of one or more datatypes. For example: array, variable length, enumeration, compound.

- **Derived:** These datatypes are created or derived from the pre-defined types.

  A simple example of creating a derived datatype is using the string datatype, H5T_C_S1, to create strings of more than one character:

```
hid_t strtype;                    /* Datatype ID */
      herr_t status;

      strtype = H5Tcopy (H5T_C_S1);
      status = H5Tset_size (strtype, 5); /* create string of length 5 */
```

Figure 5.1 shows the HDF5 pre-defined datatypes. Some of the HDF5 predefined atomic datatypes are listed in Figures 5.2a and 5.2b.

In this tutorial, we consider only HDF5 predefined integers.

For further information on datatypes, see The Datatype Interface (H5T) in the —*HDF5 User's Guide*, in addition to the Datatypes tutorial topic.

Fig 5.1 *HDF5 datatypes*

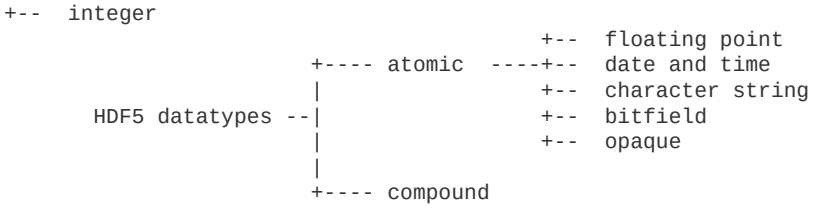

Fig. 5.2a *Examples of HDF5 predefined datatypes*

Datatype	Description
H5T_STD_I32LE	Four-byte, little-endian, signed, two's complement integer
H5T_STD_U16BE	Two-byte, big-endian, unsigned integer
H5T_IEEE_F32BE	Four-byte, big-endian, IEEE floating point
H5T_IEEE_F64LE	Eight-byte, little-endian, IEEE floating point
H5T_C_S1	One-byte, null-terminated string of eight-bit characters

Fig. 5.2b *Examples of HDF5 predefined native datatypes*

Native Datatype	Corresponding C or FORTRAN Type
C:	
H5T_NATIVE_INT	int

H5T_NATIVE_FLOAT	float
H5T_NATIVE_CHAR	char
H5T_NATIVE_DOUBLE	double
H5T_NATIVE_LDOUBLE	long double
FORTRAN:	
H5T_NATIVE_INTEGER	integer
H5T_NATIVE_REAL	real
H5T_NATIVE_DOUBLE	double precision
H5T_NATIVE_CHARACTER	character

Datasets and Dataspaces

A dataspace describes the dimensionality of the data array. A dataspace is either a regular N-dimensional array of data points, called a simple dataspace, or a more general collection of data points organized in another manner, called a complex dataspace. Figure 5.3 shows HDF5 dataspaces. In this tutorial, we only consider simple dataspaces.

Fig 5.3 *HDF5 dataspaces*

```
+-- simple
       HDF5 dataspaces --|
                         +-- complex
```

The dimensions of a dataset can be fixed (unchanging), or they may be unlimited, which means that they are extensible. A dataspace can also describe a portion of a dataset, making it possible to do partial I/O operations on selections.

Property Lists

Property lists are a mechanism for modifying the default behavior when creating or accessing objects. For more information on property lists see the Property List tutorial topic.

The following property lists can be specified when creating a dataset:

- Dataset Creation Property List
  When creating a dataset, HDF5 allows the user to specify how raw data is organized and/or compressed on disk. This information is stored in a dataset creation property list and passed to the dataset interface. The raw data on disk can be stored contiguously (in the same linear way that it is organized in memory), partitioned into chunks, stored externally, etc. In this tutorial, we use the default dataset creation property list (contiguous storage layout and no compression). For more information about dataset creation property lists, see The Dataset Interface (H5D) in the *—HDF5 User's Guide*.

- Link Creation Property List
  The link creation property list governs creation of the link(s) by which a new dataset is accessed and the creation of any intermediate groups that may be missing.

- Dataset Access Property List
  Dataset access property lists are properties that can be specified when accessing a dataset.

Steps to Create a Dataset

To create an empty dataset (no data written) the following steps need to be taken:

1. Obtain the location identifier where the dataset is to be created.

2. Define or specify the dataset characteristics:

    a. Define a datatype or specify a pre-defined datatype.

    b. Define a dataspace.

    c. Specify the property list(s) or use the default.

3. Create the dataset.

4. Close the datatype, the dataspace, and the property list(s) if necessary.

5. Close the dataset.

In HDF5, datatypes and dataspaces are independent objects which are created separately from any dataset that they might be attached to. Because of this, the creation of a dataset requires the definition of the datatype and dataspace. In this tutorial, we use the HDF5 predefined datatypes (integer) and consider only simple dataspaces. Hence, only the creation of dataspace objects is needed.

High Level APIs

The High Level HDF5 Lite APIs (H5LT) include functions that simplify and condense the steps for creating datasets in HDF5. The examples in the following section use the standard APIs. For a quick start you may prefer to look at the HDF5 Lite APIs at this time.

If you plan to work with images, please look at the High Level HDF5 Image APIs (H5IM), as well.

Programming Example

Description

See HDF5 Introductory Examples for the examples used in the Learning the Basics tutorial.

The example shows how to create an empty dataset. It creates a file called `dset.h5` in the C version (`dsetf.h5` in Fortran), defines the dataset dataspace, creates a dataset which is a 4x6 integer array, and then closes the dataspace, the dataset, and the file:

For details on compiling an HDF5 application: [Compile Information]

Remarks

H5S_CREATE_SIMPLE creates a new simple dataspace and returns a dataspace identifier.
H5S_CLOSE releases and terminates access to a dataspace.

Example code:

C:
```
    dataspace_id = H5Screate_simple (rank, dims, maxdims);
    status = H5Sclose (dataspace_id );
```

FORTRAN:
```
    CALL h5screate_simple_f (rank, dims, dataspace_id, hdferr, maxdims=max_dims)
         or
    CALL h5screate_simple_f (rank, dims, dataspace_id, hdferr)

    CALL h5sclose_f (dataspace_id, hdferr)
```

H5D_CREATE creates an empty dataset at the specified location and returns a dataset identifier.
H5D_CLOSE closes the dataset and releases the resource used by the dataset. This call is mandatory.

Example code:

C:
```
   dataset_id = H5Dcreate(file_id, "/dset", H5T_STD_I32BE, dataspace_id,
                          H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);
   status = H5Dclose (dataset_id);
```

FORTRAN:
```
   CALL h5dcreate_f (loc_id, name, type_id, dataspace_id, dset_id, hdferr)
   CALL h5dclose_f (dset_id, hdferr)
```

Note that if using the pre-defined datatypes in FORTRAN, then a call must be made to initialize and terminate access to the pre-defined datatypes:

```
CALL h5open_f (hdferr)
  CALL h5close_f (hdferr)
```

H5_OPEN must be called before any HDF5 library subroutine calls are made;
H5_CLOSE must be called after the final HDF5 library subroutine call.

See the programming example for an illustration of the use of these calls.

File Contents

The contents of the file `dset.h5` (`dsetf.h5` for FORTRAN) are shown in **Figure 5.4** and **Figures 5.5a** and **5.5b**.

Figure 5.4 *Contents of `dset.h5` (`dsetf.h5`)*

Figure 5.5a *`dset.h5` in DDL*

```
HDF5 "dset.h5" {
GROUP "/" {
   DATASET "dset" {
      DATATYPE { H5T_STD_I32BE }
      DATASPACE { SIMPLE ( 4, 6 ) / ( 4, 6 ) }
      DATA {
         0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0
      }
   }
}
}
```

Figure 5.5b *`dsetf.h5` in DDL*

```
HDF5 "dsetf.h5" {
GROUP "/" {
   DATASET "dset" {
      DATATYPE { H5T_STD_I32BE }
      DATASPACE { SIMPLE ( 6, 4 ) / ( 6, 4 ) }
      DATA {
         0, 0, 0, 0,
         0, 0, 0, 0,
         0, 0, 0, 0,
         0, 0, 0, 0,
         0, 0, 0, 0,
         0, 0, 0, 0
      }
   }
}
}
```

Note in Figures 5.5a and 5.5b that `H5T_STD_I32BE`, a 32-bit Big Endian integer, is an HDF atomic datatype.

Dataset Definition in DDL

The following is the simplified DDL dataset definition:

Fig. 5.6 *HDF5 Dataset Definition*

```
<dataset> ::= DATASET "<dataset_name>" { <datatype>
                                         <dataspace>
                                         <data>
                                         <dataset_attribute>* }

      <datatype> ::= DATATYPE { <atomic_type> }

      <dataspace> ::= DATASPACE { SIMPLE <current_dims> / <max_dims> }

      <dataset_attribute> ::= <attribute>
```