

For example, consider a 5x3 array of data of type H5T_STD_I16BE. H5T_STD_I16BE is a built-in atomic datatype of class H5T_INTEGER. Each element of such an array is a 16-bit, big-endian integer value. This dataset has an atomic datatype.

Now consider a 4 row dataset of class H5T_ARRAY where each element of the dataset is a 5x3 array of data of type H5T_STD_I16BE. This dataset has a composite datatype. Each of the 4 elements of the dataset is an array, in this case, an array of 16-bit, big-endian integer values. H5T_STD_I16BE is the "base datatype" of each of the 4 arrays.

Now consider a 4 row dataset of class H5T_COMPOUND with one integer member, one float member and one string member. This dataset has a composite datatype. Each of the 4 elements of this dataset is a structure with members of three different datatypes.

In HDF5 it is possible to nest datatypes to any depth. For example, you can create an H5T_ARRAY dataset where each element is an H5T_COMPOUND dataset where the members are H5T_ARRAY datasets of H5T_STD_I16BE, H5T_STD_I32BE and other datatypes.

Igor does not support indefinitely nested datasets. It supports only the following:

- Atomic datasets of almost any type.
- Array datasets where the base type is integer, float, string, bitfield, opaque or enum.
- Compound datasets where the member's base type is integer, float, string, bitfield, opaque, enum or reference.
- Compound datasets with array members where the base type of the array is integer, float, string, bitfield, opaque, enum or reference.
- Variable-length datasets where the base type is integer or float.

HDF5 Compound Data

This is an advanced feature that most users will not need.

In an HDF5 compound dataset, each element of the dataset consists of a set of named fields, like an instance of a C structure. Loading compound datasets is problematic because their structure can be arbitrarily complex.

A compound data set may contain a collection of disparate datatypes, arrays of disparate datatypes, and sub-compound structures.

HDF5LoadData can load either a single member from a compound dataset into a single wave, or it can load all members of the compound dataset into a set of waves. However, if a member is too complex, HDF5LoadData can not load it and returns an error. For the most part, "too complex" means that the member is itself compound (a sub-structure).

You instruct the HDF5LoadData operation to load a single member from each element of the compound dataset by using the /COMP flag with a mode parameter set to one and with the name of the member to load. The member must be an atomic datatype or an array datatype but can not be another compound datatype (see HDF5 Nested Datatypes for details). HDF5LoadData creates an Igor wave with a data type that is compatible with the datatype of the HDF5 dataset. The name of the wave is based on the name of the dataset or attribute being loaded or on the name specified by the /N flag.

You instruct HDF5LoadData to load all members into separate waves by omitting the /COMP flag or specifying /COMP={0,""}. The names of the waves created consist of a base name concatenated with a possibly cleaned up version of the member name. The base name is based on the name of the dataset or attribute being loaded or on the name specified by the /N flag.

Although you can load an HDF5 dataset with a compound datatype, Igor currently provides no way to write a datatype with a compound datatype.