HDF5 Compression

Igor can use compression when writing datasets to HDF5 files. Compressed datasets usually take less space on disk but more time to write and more time to read.

The amount of disk space saved by compression depends on the size and nature of your data. If your data is very noisy, compression may save little disk space or even increase the amount of disk space required. If your data is nearly all zero or some other constant value, compression may save 99 percent of disk space. With typical real-world data, compression may save from 10 to 50 percent of disk space.

The time required to save a compressed dataset is typically in the range of 2 to 10 times the time required to save an uncompressed dataset. The time required to load a compressed dataset is typically 2 times the time required to load an uncompressed dataset. These times can vary widely depending on your data.

Compression may be performed when you

- Save an HDF5 packed experiment file
- Save a dataset or group using the HDF5 Browser
- Call the HDF5SaveData operation
- Call the HDF5SaveGroup operation

To enable compression Igor must provide certain parameters to the HDF5 library:

- The level of compression for GZIP from 0 to 9
- Whether shuffling should be performed or not
- The chunk size to use for each dimension of the wave being saved

Compression parameters for a given dataset are set when the dataset is created and can not be changed when appending to an existing dataset.

There are several ways through which Igor obtains these parameters:

- From the HDF5SaveData operation /GZIP and /LAYO flags
- From the HDF5SaveGroup, SaveExperiment, and SaveData operation /COMP flags
- From the SaveGraphCopy, SaveTableCopy, and SaveGizmoCopy operation /COMP flags
- From the HDF5SaveDataHook (see **Using HDF5SaveDataHook** on page II-215 for details)
- Via HDF5 default compression for manual saves (see **HDF5 Default Compression** on page II-214 for details)

HDF5 supports compression of datasets through three categories of filters:

- Internal filters: Shuffle, Fletcher32, ScaleOffset, and NBit

  Internal filters are implemented in the HDF5 library source code.
- External filters: GZIP, SZIP

  External filters are linked to the HDF5 library when the library is compiled.
- Third-party filters

  Third-party filters are detected and loaded by the HDF5 library at runtime.

Igor Pro currently supports the following HDF5 filters:

- GZIP
- SZIP for reading only - not supported for writing
- Shuffle

  Shuffle reorders the bytes of multi-byte data elements and can result in higher compression ratios.