- Whether default compression is enabled or disabled
- The minimum size a wave must be before compression is used
- A compression level from 0 (no compression) to 9 (max)
- Whether you want to enable shuffle (an optional process before compression)

You can programmatically determine the default compression currently in effect like this:

```
Print IgorInfo(14)      // Print default compression settings
```

Once you have turned default compression on, it applies to saving HDF5 packed experiment files via the user interface, namely via the Data Browser Save Copy button, via the HDF5 Browser Save Waves and Save Data Folder buttons, and via the following File menu items: Save Experiment, Save Experiment As, Save Experiment Copy, Save Graph Copy, Save Table Copy, and Save Gizmo Copy.

The HDF5SaveDataHook function, if it exists, can override default compression settings.

Igor applies default compression only to numeric waves, not to text waves or other non-numeric waves nor to numeric waves with fewer than the number of elements specified by the settings.

Default compression uses chunk sizes equal to the size of each dimension of the wave. Such chunk sizes mean that the entire wave is written using chunked layout as one chunk. This is fine for datasets that are to be read all at once.

Igor's HDF5 default compression is sufficient for most purposes. If necessary, for example if you intend to read subsets rather than the entire dataset at once, you can override it using the HDF5SaveData operation or an HDF5SaveDataHook function.

Using HDF5SaveDataHook

HDF5SaveDataHook is a user-defined function that Igor calls to determine what kind of compression, if any, should be applied when saving a given wave as a dataset.

Support for HDF5SaveDataHook was added in Igor Pro 9.00.

NOTE: HDF5SaveDataHook is an experimental feature for advanced users only. The feature may be changed or removed. If you find it useful, please let us know, and send your function and an explanation of what purpose it serves.

The hook function takes a single HDF5SaveDataHookStruct parameter containing input and output fields. The version, operation, and waveToSave fields are inputs. The gzipLevel, shuffle, and chunkSizes fields are outputs.

The main use for HDF5SaveDataHook is to provide a way by which you can specify chunk sizes on a wave-by-wave basis taking into account the wave dimension sizes and your tradeoff of disk space versus save time. The HDF5SaveData operation allows you to specify chunk sizes via the /LAYO flag. However, the other operations that support HDF5 compression through the /COMP flag (HDF5SaveGroup, SaveExperiment, SaveData, SaveGraphCopy, SaveTableCopy, and SaveGizmoCopy) always save datasets as one chunk as does HDF5 default compression. HDF5SaveDataHook allows you to change that, though in most cases there is no need to.

If the operation field is non-zero, then the hook function is being called from the HDF5SaveData or HDF5SaveGroup operations. You should respect the compression settings specified to those operations unless there is good reason to override them.

To set the compression parameters for the specified wave, set the output fields and return 1. To allow the save to proceed without altering compression, return 0.

Here is an example:

```
ThreadSafe Function HDF5SaveDataHook(s)
   STRUCT HDF5SaveDataHookStruct& s
   // Print "Global ThreadSafe HDF5SaveDataHook called"// For debugging only
   // return 0             // Uncomment this to make this hook a NOP

   if (s.version >= 2000)  // Structure version is 1000 as of this writing
      return 0             // The structure changed in an incompatible way
   endif

   if (s.operation != 0)   // Called from HDF5SaveData or HDF5SaveGroup?
      return 0             // Respect their settings unless we have
   endif                   // good reason to change them

   // Set compression parameters only for numeric waves
   int isNumeric = WaveType(s.waveToSave) == 1
   if (!isNumeric)
      return 0
   endif

   // Set compression parameters only if the total number of elements
   // in the wave exceeds a particular threshold
   int numElements = numpnts(s.waveToSave)
   if (numElements < 10000)
      return 0
   endif

   // Set compression parameters
   s.gzipLevel = 5         // 0 (no compression) to 9 (max compression)
   s.shuffle = 0           // 0=shuffle off, 1=shuffle on
   s.chunkSizes[0] = 200   // These values are arbitrary. Igor clips them.
   s.chunkSizes[1] = 200   // In a real function you would have some more
   s.chunkSizes[2] = 50    // systematic way to choose them.
   s.chunkSizes[3] = 50

   // Indicate that we want to set compression parameters for this wave
   return 1
End
```

The HDF5SaveDataHook function must be threadsafe. If it is not threadsafe, Igor will not call it.

The HDF5SaveDataHook function is called when a dataset is created and not when appending to an existing dataset.

The HDF5SaveDataHook function must not alter the state of Igor. That is, it must have no side effects such as creating waves or variables, modifying waves or variables, or killing waves or variables. It is called when HDF5SaveData is called, when HDF5SaveGroup is called, and also when Igor is saving an HDF5 packed experiment file. If you change the state of Igor in your HDF5SaveDataHook function, this may cause file corruption or a crash.

It is possible but not recommended to define more than one HDF5SaveDataHook function. For example, you can define one in the ProcGlobal context, another in a regular module and a third in an independent module. If you do this, each function is called until one returns 1. The order in which the functions are called is not defined.

We recommend that you define your HDF5SaveDataHook function in an independent module so that it can be called when normal procedures are in an uncompiled state. Otherwise, if you save an HDF5 packed experiment while procedures are not compiled, for example because of an error, your HDF5SaveDataHook function will not be called and all datasets will be saved with default compression or uncompressed.

For testing purposes, you can prevent Igor from calling any HDF5SaveDataHook function by executing:

```
SetIgorOption HDF5SaveDataHook=0
```