Next we will learn how to write an Igor procedure to access an HDF5 file programmatically. Before you start that, feel free to play with the HDF5 Browser on your own HDF5 files. Start by choosing Data→Load Waves→New HDF5 Browser.

Igor can handle most HDF5 files that you will encounter, but it can not handle all possible HDF5 files. If you receive an error while examining your own files, it may be because of a bug or because Igor does not support a feature used in your file. In this case you can send the file along with a brief explanation of the problem to support@wavemetrics.com and we will determine whether the problem is a bug or just a limitation.

Loading HDF5 Data Programmatically

Igor includes a number of operations and functions for accessing HDF5 files. To get an idea of the range and scope of the operations, click the link below. Then return here to continue the guided tour.

HDF5 Operations and Functions on page II-197

In this section, we will load an HDF5 dataset using a user-defined function.

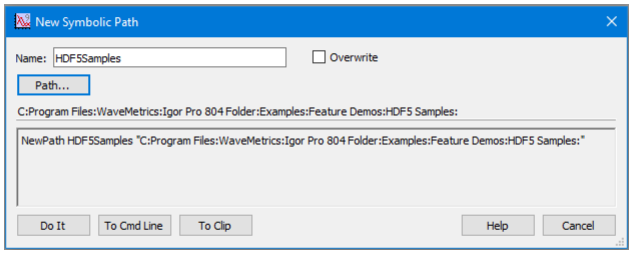
1. **Choose File→New Experiment.**

   You will be asked if you want to save the current experiment. Click No (or Yes if you want to save your current work environment).

   This closes the current experiment, killing any waves and windows.

2. **Choose Misc→New Path and create an Igor symbolic path named HDF5Samples and pointing to the HDF5 Samples directory which contains the TOVSB1NF.h5 file that we used in the preceding section.**

   The New Path dialog will look something like this:



New Symbolic Path

Name: HDF5Samples

☐ Overwrite

C:Program Files:WaveMetrics:Igor Pro 804 Folder:Examples:Feature Demos:HDF5 Samples:

```
NewPath HDF5Samples "C:Program Files:WaveMetrics:Igor Pro 804 Folder:Examples:Feature Demos:HDF5 Samples:"
```

   Click Do It to create the symbolic path.

   An Igor symbolic path is a short name that references a specific directory on disk. We will use this symbolic path in a command that opens an HDF5 file.

   In HDF5, you must first open a file or create a new file. The HDF5 library returns a file ID that you use in subsequent calls to the library. When you are finished, you must close the file. The following example illustrates this three step process.

3. **Choose Windows→Procedure Window and paste the following into the Procedure window:**

```
Function TestLoadDataset(datasetName)
    String datasetName              // Name of dataset to be loaded

    Variable fileID                 // HDF5 file ID will be stored here

    Variable result = 0             // 0 means no error

    // Open the HDF5 file
    HDF5OpenFile /P=HDF5Samples /R /Z fileID as "TOVSB1NF.h5"
    if (V_flag != 0)
        Print "HDF5OpenFile failed"
        return -1
    endif

    // Load the HDF5 dataset
```

```
	HDF5LoadData /O /Z fileID, datasetName
	if (V_flag != 0)
		Print "HDF5LoadData failed"
		result = -1
	endif

	// Close the HDF5 file
	HDF5CloseFile fileID

	return result
End
```

Read through the procedure. Notice that we use the symbolic path HDF5Samples created above as a parameter to the HDF5OpenFile operation.

The /R flag was used to open the file for read-only so that we don't get an error if the file can not be opened for writing.

The /Z flag tells HDF5OpenFile that, if there is an error, it should set the variable V_flag to a non-zero value but return to Igor as if no error occurred. This allows us to handle the error gracefully rather than having Igor abort execution and display an error dialog.

HDF5OpenFile sets the local variable fileID to a value returned from the HDF5 library. We pass that value to the HDF5LoadData operation. We also provide the name of a dataset within that file to be loaded. Again we check the V_flag variable to see if an error occurred.

Finally we call HDF5CloseFile to close the file that we opened. It is important to always close files that we open. If, during development of a procedure, you forget to close a file or fail to close a file because of an error, you can close all open HDF5 files by executing:

```
HDF5CloseFile /A 0
```

Also, Igor automatically closes all open files if you choose File→New Experiment, File→Revert Experiment or File→Open Experiment or if you quit Igor.

4. **Click the Compile button at the bottom of the Procedure window.**

   If you don't see a Compile button at the bottom of the Procedure window, that is because the procedures were already automatically compiled when you deactivated the Procedure window.

   If you get a compile error then you have not pasted the right text into the Procedure window or you have entered other incorrect text. Try going back to step 1 of this section.

   Now we are ready to run our procedure.

5. **Execute the following command in the Igor command line either by typing it and pressing Enter, by copy/paste followed by Enter, or by selecting it and pressing Control-Enter (Macintosh) or Ctrl-Enter (Windows):**

   ```
   TestLoadDataset("Raster Image #0")
   ```

   At this point the procedure should have correctly executed and you should see no error messages printed in the history area (just above the command line). This history area of the command window should contain:

   ```
   Waves loaded by HDF5LoadData: Raster Image #0
   ```

   Now we will verify that the data was loaded.

6. **Choose Data→Data Browser. Then double-click the wave named 'Raster Image #0'.**

   This creates a table showing the data just loaded.

   At this point Igor's root data folder contains just one wave, 'Raster Image #0', and does not contain the corresponding palette wave. That's because we loaded the dataset as a plain dataset, not as a formal image. The HDF5LoadData operation does not know about formal images. For that we need to use HDF5LoadImage.

7. **In the Data Browser, Control-click (Macintosh) or right-click (Windows) the 'Raster Image #0' wave and choose New Image.**

   Igor creates an image plot of the wave.