This file has the original section and an edited section.

### 7.4. Dataspaces and Data Transfer (original section)

The dataspace object is also used to control data transfer when data is read or written. The dataspace of the dataset (attribute) defines the stored form of the array data, the order of the elements as explained above. When reading from the file, the dataspace of the dataset defines the layout of the source data, a similar description is needed for the destination storage. A dataspace object is used to define the organization of the data (rows, columns, etc.) in memory. If the program requests a different order for memory than the storage order, the data will be rearranged by the HDF5 Library during the H5Dread operation. Similarly, when writing data, the memory dataspace defines the source data, which is converted to the dataset dataspace when stored by the H5Dwrite call.

Item a in the figure below shows a simple example of a read operation in which the data is stored as a 3 by 4 array in the file (item b), but the program wants it to be a 4 by 3 array in memory. This is accomplished by setting the memory dataspace to describe the desired memory layout, as in item c. The HDF5 Library will transform the data to the correct arrangement during the read operation.

|  |
| --- |
|  |
|  |
| **Figure 3. Data layout before and after a read operation** |

|  |
| --- |
|  |
|  |
| **Figure 4. Moving data from disk to memory** |

Both the source and destination are stored as contiguous blocks of storage with the elements in the order specified by the dataspace. The figure above shows one way the elements might be organized. In item a, the elements are stored as 3 blocks of 4 elements. The destination is an array of 12 elements in memory (see item c). As the figure suggests, the transfer reads the disk blocks into a memory buffer (see item b), and then writes the elements to the correct locations in memory. A similar process occurs in reverse when data is written to disk.

### 7.4. Dataspaces and Data Transfer (edited section)

Read and write operations transfer data between an HDF5 file on disk and in memory. The shape that the array data takes in the file and in memory may be the same, but HDF5 also allows users the ability to represent data in memory in a different shape than in the file. If the shape of an array in the file and in memory will be the same, then the same dataspace definition can be used for both. If the shape of an array in memory needs to be different than the shape in the file, then the dataspace definition for the shape of the array in memory can be changed. During a read operation, the array will be read into the different shape in memory, and during a write operation, the array will be written to the file in the shape specified by the dataspace in the file. The only qualification is that the number of elements read or written must be the same in both the source and the destination dataspaces.

Item a in the figure below shows a simple example of a read operation in which the data is stored as a 3 by 4 array in the file (item b) on disk, but the program wants it to be a 4 by 3 array in memory. This is accomplished by setting the memory dataspace to describe the desired memory layout, as in item c. The read operation reads the data in the file array into the memory array.

|  |
| --- |
|  |
|  |
| **Figure 3. Data layout before and after a read operation** |

|  |
| --- |
|  |
|  |
| **Figure 4. Moving data from disk to memory** |

Both the source and destination are stored as contiguous blocks of storage with the elements in the order specified by the dataspace. The figure above shows one way the elements might be organized. In item a, the elements are stored as 3 blocks of 4 elements. The destination is an array of 12 elements in memory (see item c). As the figure suggests, the transfer reads the disk blocks into a memory buffer (see item b), and then writes the elements to the correct locations in memory. A similar process occurs in reverse when data is written to disk.