h5tools-matlab

The purpose of this documentation is to detail the implementation of h5tools. For information on how to use each function, see the documentation within the code using `help` or `doc`.

1 Datasets

The high-level functions for datasets are `h5tools.write` and `h5tools.read`. The implementation for these functions can be found in the "h5tools.datasets" package. The key difference between these functions and their corresponding MATLAB versions (`h5read` and `h5write`) are:

1. With MATLAB's library, you need to first create the dataset with `h5create` (which requires some understanding of HDF5 dataspaces), then write it with `h5write`. `h5tools.write` takes care of dataspaces and dataset creation behind the scenes.
2. Support for more MATLAB data types. Text (`char` and `string`), enumerated types, and compound datatypes (`table`) are directly supported, in addition to the numeric types MATLAB's high-level library implements. and `timetable`) are also directly supported, with some caveats described below. A few tricks are used to support additional types, by converting to an HDF5-compatible format and adding attributes to the dataset that guide h5tools in reading the data back into MATLAB (`logical`, `cellstr`, `struct`, `containers.Map`, `datetime`, `duration`).
3. Writing soft links (a dataset that acts as a "shortcut" to a different group or dataset within the HDF5 file) is supported with `h5tools.writelink`.

In describing each datatype's implementation, the following information is provided where relevant:

- Preprocessing: Any changes to the data's MATLAB type that occur before writing, to make the data compatible with an HDF5 datatype.
- Persistence: How the data is actually written to the HDF5 dataset.
- Postprocessing: Any processing of the data that occurs after reading into MATLAB, to make the data compatible a specific MATLAB datatype.
- Implementation: The functions used to implement writing and reading of the dataset. All data passes through `h5tools.writeDatasetByType` for writing and `h5tools.readDatasetByType` for reading. Any additional functions implementing specialized processing will be detailed here.
- Limitations: Any caveats or limitations.

1.1 Numeric

MATLAB supports direct conversion of numeric datatypes to and from HDF5 files with `h5read` and `h5write`. The advantage of h5tools is a more user-friendly interface that automates dataset creation (`h5create`).

- Preprocessing: As with all other datatypes, the dataset creation and dataspace creation is handled rather than requiring the use of `h5create`. Writing is performed with `h5read`.
- Persistence: See **Table 1** for correspondence between MATLAB numeric types and HDF5 data types.
- Postprocessing: N/A
- Implementation: `h5tools.datasets.makeMatrixDataset`
- Limitations: N/A

1.2 Char

- Preprocessing: N/A
- Persistence: Written as class `H5T_STRING` with character type `H5T_C_S1` and set `H5T_CSET_ASCII`.

- Postprocessing: N/A
- Implementation: `h5tools.datasets.makeTextDataset`, `h5read`
- Limitations: N/A

1.3 String

- Preprocessing: N/A
- Persistence: written as class `H5T_STRING` with character type `H5T_C_S1` and set `H5T_CSET_UTF8`.
- Postprocessing: N/A
- Implementation: `h5tools.datasets.makeStringDataset`, `h5read`
- Limitations: N/A

1.4 Cellstr

While **`cellstr`** is not supported, it is comparable to a **`string`** array and can be written as such.

- Preprocessing: Conversion to **`string`** array.
- Persistence: Identical to **`string`**, but with 1 attribute: *`Class`* = `cellstr`.
- Postprocessing: Conversion from **`string`** back to **`cellstr`**
- Implementation: See **`string`**.
- Limitations: N/A

1.5 Datetime

While **`datetime`** is not supported (HDF5 has a time type but I have no looked into it yet), the data can be represented as a **`string`**, as long as the format is recorded.

- Preprocessing: The "format" property is extracted, then the data is converted to **`string`**.
- Persistence: See **`string`**. Two attributes are written to the dataset: *`Class`* = `datetime` and *`Format`* = `date format`.
- Postprocessing: Conversion to **`datetime`** using the optional Format argument.
- Implementation: `h5tools.datasets.makeDateDataset`
- Limitations: If providing an array of **`datetime`** values, they must have the same format.

1.6 Enumerated types

HDF5 offers enumerated type, which is an integer datatype restricted to a set of named values. For example, an enumerated type may contain three members (e.g. “Red”, “Green”, “Blue”) and their corresponding values (0, 1, 2).

- Preprocessing: The index of each enumerated type value within the full list of enumerated types is converted to **`uint8`**.
- Persistence: written as `H5T_ENUM` with the values of each member in the enumeration as `H5T_STD_I32LE`. While enumerated types supported in HDF5, accompanying information is required to identify the class containing the enumeration when read back into MATLAB. This information is written as two attributes: *`Class=enum`* = and *`EnumClass`* = `MATLAB class` name containing enumeration (including namespaces, if applicable).
- Postprocessing: A string array of the enumeration members is read from the dataset through the datatype library (**`H5T`**), ordered by their enumeration value. If the associated class is not found on MATLAB’s path, the data will be read back in as a string array using the format **`x.y`** where **`x`** is the class name in the attributes and **`y`** is the enumeration name. If the class is found on MATLAB’s path, the string

will be evaluated (`eval`). If the enumerated type was not written with h5tools and as no class name attribute, the string array will just contain the enumeration names.

- Implementation:
- Limitations:

1.7 Logical

HDF5 has no boolean datatype so, following other successful HDF5 libraries like h5py, this is written as an enumerated type.

- Preprocessing: Conversion to `uint8`.
- Persistence: written as `H5T_ENUM` with two members (0=FALSE, 1=TRUE).
- Postprocessing: If an enumerated type dataset is read in and has two enumerated types matching those above, the values it will be treated as `logical`. The enumerated values will be converted from `uint8` to `logical`.
- Implementation:
- Limitations: N/A

1.8 Table

HDF5 supports writing tables as compound datatypes, for which there is no MATLAB support.

- Preprocessing:
- Persistence: written as `H5T_COMPOUND` with two attributes: *`Class`* `= table`, *`ColumnClasses`* `= [class1, class2,...]` of data in each column
- Postprocessing:
- Implementation: `h5tools.datasets.makeCompoundDataset`
- Limitations: Currently only numeric datatypes, string and char are supported.

1.9 Struct

- Preprocessing: The number of elements in each field will be calculated to determine the strategy used to write the information to the HDF5 file.
- Persistence: If the number of elements in each field is the same, `H5T_COMPOUND` is used, as described above, but with the identifying attribute *`Class`* `= struct`.
- Postprocessing: See above for compound data type. If mapped to attributes instead, the attributes and their values will be assigned to the fields of a `struct`, omitting the Class attribute.
- Implementation: `h5tools.datasets.makeStructDataset`, `h5tools.datasets.makeMapDataset`
- Limitations: As described above, the number of elements in each field must be the same. Multi-level structs (e.g. a `struct` where one or more of the fields are `struct`. The limitations of `table` also apply.

1.10 Containers.Map

`Containers.Map` is treated similarly to `struct`.

- Preprocessing: Conversion to `struct` with `map2struct`.
- Persistence: See `struct`. Only difference is the identifying attribute *`Class`* `= containers.Map`.
- Postprocessing: If necessary, conversion with `struct2map`.
- Implementation: `h5tools.datasets.makeStructDataset`, `h5tools.datasets.makeMapDataset`
- Limitations: See `struct` and `table`.

2 Attributes

The high-level functions for attributes are `h5tool.writeatt` and `h5tools.readatt`. Many of the classes supported by h5tools for datasets rely on attributes to ensure accurate conversion when reading in the data. Because attributes cannot have attributes, the additional datatypes available here are less extensive.

3 Soft Links

Soft links are comparable to "shortcuts" within a file system. Currently, h5tools supports writing a dataset that is a soft link to another dataset or group within the HDF5 file (`h5tools.writelink`). Arrays of soft links and soft link attributes are not currently supported, but will be added in the near future.

4 Limitations

Most of the functions within h5tools are straightforward solutions. However, some of the dataset/attribute writing functions employ tricks rather than true solutions to ensure as many MATLAB datatypes can be read and written to HDF5 files as possible. For those situations, keep the following limitations in mind:

4.1 Datasets

1. `struct` and `containers.Map`. Both of these data types mimic the natural layout of an HDF5 file (i.e they are a group containing datasets), so there are naturally difficulties writing them to a single dataset. Before writing a `struct`/`containers.Map` as a dataset, always consider making a new group and writing the members of the struct/map as individual datasets within that group. If you absolutely must have it as a dataset, a few key points:
   - `struct` is written as a dataset with HDF5's compound data type (like `table` and `timetable`). The datatypes of each field will be recorded in the attributes to facilitate reading the data back in appropriately. But, the struct must behave like a table, in that each field should have the same number of elements.
   - The contents of `containers.Map` is written as attributes of a placeholder text dataset. The advantage here is that each field can have a different number of elements. The disadvantage is that the attributes can't have attributes so the attribute limitations discussed below apply.
   - There are advantages and disadvantages to both, so consider which best suits your data. Conversion functions `struct2map` and `struct2map` are included in the "util" folder. A good rule of thumb for `struct`, `containers.Map` and `table` is to always double check when writing one containing new datatypes and ensure it's being read back in as you expect.
2. Multi-level structs (i.e. a `struct` containing another `struct`) are not supported. Instead, make a new group for the secondary `struct`.
3. Enumerated types can be read back into MATLAB appropriately, but only if the class containing the enumeration is on your search path. If the class can't be found, they will be read back in as `string`.
4. There are also several features that MATLAB's built-in functions already handle well and are not currently supported in h5tools, such as reading subsets of numeric datasets, using filters and working with remote locations.

4.2 Attributes

A number of the specialized data types above are supported by attaching attributes to the dataset with the necessary additional information to allow MATLAB to read them appropriately. Attributes can't have attributes, so the datatype support is naturally more limited.

1. The standard numeric datatypes, string, char and logical are fully supported. As above, reading `enum` is only supported if the class is on your search path.

2. Specialized MATLAB datatypes (e.g. datetime, duration, etc) cannot be written as HDF5 attributes and read identically back into MATLAB. If possible, `h5tools.writeatt` will try to convert your information to a form that can be written. For example, **`datetime`** will converted with `datestr` and written as **`string`**. The information at least exists in your file, but when you read that attribute back into MATLAB from the HDF5 file, it come back as a **`string`**. When you write a **`datetime`** as a dataset, `h5tools.write` adds an attribute specifying the class and the datetime format, so that `h5tools.read` can convert it back to **`datetime`**. This won't work for attributes as they can't have their own attributes. Other attributes that can be written but not identically read back include **`duration`** (converted to **`double`**) and **`cellstr`** (converted to **`string`**).
3. Attributes that are links to other groups/datasets have not been implemented.
4. Row vectors will be returned as columns. I haven't had a chance to figure out why this is or whether there's a fix.

If you try to write an attribute without write support, you'll get an error. If you write one without read support, a warning will be sent to the command line specifying the path within the HDF5 file, attribute name and datatype, so that you are aware.

5 Reference

Mapping of numeric datatypes to HDF5:

MATLAB type	**HDF5 Class**	**HDF5 Type**
double	H5T_FLOAT	H5T_IEEE_F64LE
single	H5T_FLOAT	H5T_IEEE_F32LE
uint8	H5T_INTEGER	H5T_STD_U8LE
int8	H5T_INTEGER	H5T_STD_I8LE

6 Resources

- MATLAB: See their documentation for the high-level functions as well as the low-level library.
- NeurodataWithoutBorders: nwb-matlab has HDF5 support for a number of data types. Although their HDF5 functions are specific to NWB, it's still a great resource for learning about HDF5 I/O.
- h5py: an excellent Python library for HDF5 – if your project isn't tied to MATLAB, consider using this instead. For example, idea to make **`logical`** an enumerated type was borrowed from h5py.