HDFds – Conventions to facilitate data sharing using HDF5

Version 0.5. 2013-01-07

INCF electrophysiology HDF5 working group

Introduction

The HDF5 file format is becoming increasingly popular for storing scientific data. To allow data stored in HDF5 files to be easily shared within a domain, as is encouraged by many data sharing initiatives, the data and metadata must be stored in HDF5 files using domain specific standards. However, it is difficult to specify such standards for HDF5 files because unlike XML, which have methods to constrain file layouts (XML DTD and XML schema), there is no standard method for describing how information should be stored in HDF5 files. This document presents a partial solution to this problem by giving conventions that allows specifying schemas for storing data and metadata in HDF5 files. We call these conventions "HDFds" which stands for "Hierarchical Data Format – data sharing". The conventions were originally created for neuroscience electrophysiology data, but the methods are general and can be used for other domains as well.

In HDF5 data are stored using multiple dimensional arrays (called "datasets"), and container objects (called "groups") that function like directories in a file system. Both groups and datasets are also called "nodes". Each node can have attributes that contain metadata. HDFds consists of conventions that provide a standard way of describing data and metadata in HDF5 files. The conventions are comprised of the following parts:

1. Specifying schemata for HDF5 nodes
2. Storing metadata and specifying schemata for metadata
3. References to datasets and external files
4. Location for storing metadata
5. Associating metadata to data

Each part (after the first) depends on previous parts.

Part 1: Specifying schemata for HDF5 nodes

To allow specifying schemata for HDF5 nodes an association must be made between one or more external documents containing schemata and the HDF5 file. To do this, HDFds uses an attribute named "schema" at the root node of the HDF5 file. It is assigned to a list of strings, each of which is an abbreviation, followed by a space, then a URL. An example list containing two such strings is below.

```
["ax http://url_x", "ay http://url_y"]
```

(In this example, "ax" and "ay" are abbreviations; URLs start with "http").

Each URL is associated with a schema that specifies some aspect of the data layout. The abbreviations can be used subsequently in the HDF5 file to reference specific entities (types of data or metadata) described in the schema. The schema should specify how the referenced entities are stored in HDF5. This is shown schematically in Figure 1.

In HDF5 file:
Root
.attribute[“schema”] = `["ax http://url_x", "ay http://url_y"]`

Groups or dataset:
.attribute[“schema”] = `"ax:foo ay:bar"`

Documentation located at: http://url_x:
foo: - description of entity foo and how it is stored in HDF5.

Document located at: http://url_y:
bar: - description of entity bar and how it is stored in HDF5.

Figure 1: Specifying schema for HDF5 groups and data sets.

In Figure 1, there are two schemas referenced which are described in documents located at url_x and url_y. They are associated with abbreviations “ax” and “ay”. Entities described in the documents at these URLs can be referenced by the “schema” attribute in groups or datasets later in the file. In Figure 1, both “ax:foo” and “ay:bar” refer to schema which describe the layout of the two referenced entities. Referencing the schema for two entities in the same node could be useful if they have non-overlapping specifications. In most cases, it is expected that a HDF5 node would reference the schema for only one entity. The manner in which data and metadata layouts are described in the documents referenced by the URL is left open because there are multiple ways of doing that. However, the description should provide sufficient information to enable writing a computer program that can load and use the data in a manner that makes sense for the domain.

The use of abbreviations defined at the top of the file and reference later in the file is intended to be similar to how XML namespaces can be defined and referenced in an XML document.

To reference entities described in this standard, a string like the following should be included in the list assigned to the root node “schema” attribute:

```
"hdfds http://URL_to_this_document"
```

In the HDF5 file, the string “`URL_to_this_document`” would be replaced by a URL pointing to this document, once it is no longer a draft and is available online. “hdfds” is the abbreviation which can be referenced later in the HDF5 file. A different abbreviation could be used as long as the definitions and references match. In this document, “hdfds” will be used in the examples.

Part 2: Storing metadata and specifying schema for metadata

Motivation
HDF5 provides a mechanism, "attributes", for including metadata in a HDF5 file. But plain HDF5 attributes have several limitations. First, there is no standard way of specifying what the metadata should contain (a schema). Second, HDF5 attributes can store a single value or an array of values, but not a structure containing mixed types. This capability is useful, for example, to specify the units (string) associated with a numeric value, or to include a link to an ontology. To overcome these limitations, HDFds provides a standard for storing metadata that allows specification of a schema and the storage of arbitrary structures containing mixed types.

Storage of metadata
In HDFds, metadata that are required by the schema are stored in string datasets or string attributes using JSON encoding (described at: http://www.json.org/). This allows storing an arbitrary hierarchical structure of information using a collection of key–value pairs and arrays. Libraries for using JSON are available for most languages, including MatLab, Python and JavaScript.

The JSON structure used to store metadata in HDFds is an array of collections of key-value pairs. In JSON square brackets [ ] represent arrays, and braces { } denote collections of key value pairs. Using these symbols, the JSON structure used to store metadata in HDFds is shown schematically below:

```
[ {"schema":"ax:baz", …}, {"schema":"ay:entity_name", …}, … ]
```

The elements of the array above (items in braces { } containing collections of key-value pairs) that specify metadata. In this standard, the elements will be called "metadata entities". In HDFds in the top level of the JSON structure for a metadata entity, a key named "schema" should be included that specifies the location of documentation that describes the structure and semantics for the metadata. An example is shown in Figure 2:

```
{
      "schema": "ax:baz",
      "sampling_rate": { "value": 1000, "units": "Hz" },
      "units" : "mV",}
      "value_mapping": { "minValue": -100, "scale_factor": 0.012}
}
```

Figure 2: Example JSON encoded metadata.

In the above, the value for the "schema" key references (using the notation described in Part 1) documentation for the metadata. "baz" is the name of a metadata entity described in the documentation located at the URL associated with abbreviation "ax". This allows different metadata entities to be included in the file, while providing a consistent way of documenting them.

Part 3: References to datasets and external files.

Since HDF5 does not specify a standard format for metadata, it also does not specify a way of incorporating references to nodes or externals files within metadata. HDFds uses the following conventions for referencing file components and external files in the JSON encoded metadata.

a) *Referencing internal nodes*. In HDFds, nodes in the HDF5 file are referenced using a string containing a path to the node. The path can be absolute (starting with slash "/") or relative (in the same manner as used in file systems, i.e. starting with a local node or a dot-slash "./" followed by the local node name to reference a local node; or a double dot-dash "../", to "go up" one level and reference nodes closer to the root). Such references can be used to associate metadata to a node or relate two nodes, for example, to indicate that one dataset contains the source for a type of analysis and another contains the result.

b) *Referencing parts of a data set*. HDFds uses 0-based, array indexing notation, with brackets appended to a data set path, to specify values within a dataset. For example, if "foo" is a dataset containing a 2-D array, then "foo[2]" specifies the third row.

c) *Referencing external files.* HDFds uses the prefix syntax "file://" to reference a file external to the HDF5 file. The reference is done using the assumption that the HDF5 file is part of a collection of files that are organized into a structure on the file system that is maintained. Having a standard for external file references, allows adding metadata to a file which references external files and nodes within external files. If the file is an HDF5 file, components within the file can be referenced by appending a path to the file name. Example references are:

```
file://file_name.pdf
file://file_name.h5/path_within_file
file://directory/file_name.hdf5/path_within_file
```

Part 4: Location for storing metadata.

To enable locating the JSON encoded metadata, HDFds designates particular locations where the metadata is stored:

a) Global metadata (that is, metadata which applies to the entire HDF5 file) are stored either: in an string attribute of the root named "global_settings", or as the data in a string dataset named "global_settings" located directly under the root, or as the data in a string dataset having any name, located anywhere in the file, with attribute "schema" having value: "hdfds:global_settings". There should be only location in the file used to specify global metadata.

b) Metadata that is associated with local data, and not the entire file are stored either: in an attribute named "settings" which is assigned to a node (either group or dataset) or as the data in a string dataset named "settings" or as the data in a string dataset with any name, with attribute "schema" set to "hdfds:settings".

Part 5: Associating metadata to data

With HDF5 attributes, the association of metadata to dataset is straightforward: mainly attributes that are assigned to a dataset are associated with that dataset, and attributes assigned to a group are associated with the nodes contained in the group. The HDFds encoded metadata described above, can be used in the same way, with metadata stored as data in a string dataset (rather than a node attribute), treated as though it was assigned to the parent group.

If the same metadata entity is specified in multiple locations (for example, local and global) that apply to a particular node the information given is combined, with any conflicts resolved in favor of the most specific (local) definition. For example, if the variable "sampling_rate" was specified as 1000 in global metadata, but 2000 in more local metadata, the local value would take precedence for data that both metadata entities apply to.

While this normal mechanism of associating metadata to data is often sufficient, HDF5 has no standard way to associate metadata to subsets of datasets in a group or to parts of a dataset. This capability could be useful, for example, if each row of a 2-D array contained data from an individual sensor that is associated with unique metadata. HDFds provides a standard method to associate metadata with specific datasets in a group or parts of datasets using the JSON encoded schema shown below.

```
1. { "schema": "hdfds:source_map",
2.   "source_labels" : {"<dataset_name>" : ["<source_name>", …], …},
3.             "dataInfo" : [ <metadata_map>, … ] }

4. <metadata_map> = { "sources": [ <source_name>, … ],
5.                    "metadata": [ <metadata>, … ] }

6. <metadata> = { "schema": "<schema_name>", …rest of schema… }
```

Figure 3: Associating unique metadata with rows in a dataset.

Figure 3 uses a BNF-like notation to describe the JSON structure. Identifiers in angle brackets, < >, are placeholders that would be replaced by a name (<dataset_name>, <source_name>, <schema_name>) or by the structure specified in subsequent lines. Square brackets, [ ], indicate an array containing one or more entities of the specified type. Braces, { }, indicate a collection of key-value pairs. Three dots (…) indicate possible repetition of the previous item.

In brief, the structure in Figure 3 operates as follows:
Line 1. The schema is identified by the value of key "schema" being set to "hdfds:source_map".
Line 2. Each row in every dataset is associated with a unique source name.
Line 3. Multiple, <metadata_map>s can be specified. Each associates an array of source names to an array of <metadata> elements (lines 4 and 5).
Line 6. Each <metadata> element indicates a structure containing metadata. The "schema" key is set to an identifier that is used to locate documentation for the metadata structure.