h5edit User Guide

Document Version 1.3.0

June 4, 2014

This document describes how to use the `h5edit` tool. `h5edit` can currently be used to modify the attributes of an existing HDF5 file without resorting to technical programming.

For more information about The HDF Group, see www.hdfgroup.org.

Contents

1. Introduction to h5edit

The `h5edit` tool is an HDF5 file editor. It supports commands to modify the contents of an existing HDF5 file. It enables HDF5 users to modify an HDF5 file without resorting to technical programming. Its intent is for small scale modification of a file. Data intensive changes to a file, such as those involving hundreds of data points, may not be executed at an efficient speed with this tool.

Currently, this tool can only be used to modify attributes.

`h5edit` can be used in two different ways: from a **command line** with the arguments typed out or from a command line with the arguments pulled from a **command file**. Simple and short commands might be entered on the command line. More complicated arguments should be put in a command file. See the "h5edit Command-line Syntax" section below for more information.

The following is a brief description of the way `h5edit` works.

- The command is executed from the command line.
- `h5edit` opens the specified HDF5 file.
- Assuming the file has been opened successfully and if the atomicity level is set to yes or inc, `h5edit` will then make a backup copy of the file.
- The operations specified on the command line are executed.
    - `h5edit` can operate on only one file at a time.
    - Several operations can be specified with a single command-line statement.
- The changes made to the HDF5 file are written to disk (flushed) according to the atomicity level.
    - One level means the changes are either all written if all operations were successful or none are written if any operations were unsuccessful.
    - Another level means the successfully made changes are written and the unsuccessfully made changes are not written.
    - A third level means the successfully made changes are written to disk before the next operation listed on the command line is started.

For more information, see "Definition of the `h5edit` Command Language".

1.1. h5edit Command-line Syntax

h5edit is run from a command line. There are three variations of the command-line syntax. These are listed below. The elements that are used are described in the rest of this document.

Command Line Only Syntax:

```
h5edit [-c | --command] command_name command_parameters [command_name
command_parameters] --atomic atomic_level hdf5_file_name
```

Command File Syntax:

```
h5edit --command-file command_file_name --atomic atomic_level
hdf5_file_name
```

Help Syntax:

```
h5edit [-h | --help]
```

The `h5edit` command-line options are described below.

Option	Comments
`--command`	Use this option to specify a command. Each command executes an operation that will be performed on the specified HDF5 file. Currently, `h5edit` can perform the following operations with attributes: create, copy, delete, modify, and rename. The commands and their parameters are described in the "Commands and Command Parameters" section below.
`--atomic`	Use this option to specify the atomicity of the operation. With this option, users can customize how changes are written to disk. See the "Atomicity Levels" section on page 10 for more information. See the "Data Recovery" section on page 11 for more information on how a backup copy of the HDF5 file might be used.
`--command-file`	Use this option to specify a command file that `h5edit` should use when operating on an HDF5 file. The only parameter for this option is `command_file_name` which specifies the name of the command file. Command files may hold a number of operations. For example, a command file could specify a list of attributes to be created or deleted. For more information, see the "Using a Command File" section on page 11.

1.2. Commands and Command Parameters

The `--command` command-line option is used to specify a command which executes an operation. The commands are `CREATE`, `COPY`, `DELETE`, `MODIFY`, and `RENAME`. The general format of each `--command` instance is:

```
--command command_name command_parameters
```

where

`--command` is the command-line option,

`command_name` is one of the operations that `h5edit` can perform, and

`command_parameters` is one or more parameters that are used in the operation.

A number of `command_name command_parameters` pairs can be used if the pairs will all operate on a single HDF5 file. `Each command_name command_parameters` pair must be terminated with a semicolon. The command line below shows an example with two `command_name` values (`DELETE`):

```
$ h5edit --command "DELETE /attr1; DELETE /attr2;" mydata.h5
```

The possible values of the `command_name` parameter and the `command_parameters` parameter are defined in the subsections below.

1.2.1. The CREATE Command and Parameters

The `CREATE` command and parameters can be used to create a new attribute with certain characteristics. The attribute is attached to a specified dataset or group in the file. When the `CREATE` command is used, the following is the syntax:

```
CREATE <new_attribute_name> <attribute_definition >;
```

The syntax elements are described in the table below.

Element	Comments
`CREATE`	The command name. Upper case only.
`<new_attribute_name>`	The complete name of the attribute. For more information, see the "Attribute Names" section on page 13.
`<attribute_definition>`	Describes the datatype, dataspace, and data value of the attribute. For more information, see the "Attribute Definitions" section on page 14.
`;`	A terminator. Semicolons are used at the end of each command and parameter set.

The following is an example which uses the CREATE command from the command line.

```
$ h5edit --command "CREATE /sta1/month12/temperature {DATATYPE
H5T_FLOAT_NATIVE DATASPACE SCALAR DATA {-40.0}};" greenland.h5
```

The name of the new attribute is temperature. The complete name is /sta1/month12/temperature. The datatype is H5T_FLOAT_NATIVE. The dataspace is SCALAR. The data value of the attribute is -40.0. The new attribute will be located in greenland.h5.

The following is a simpler version of the command line above.

```
$ h5edit --command "CREATE /sta1/month12/temperature {H5T_FLOAT_NATIVE
SCALAR {-40.0}};" greenland.h5
```

The difference with the first example above is that the keywords have been left out.

The following is a much simpler example of using the CREATE command on the command line.

```
$ h5edit --command "CREATE /sta1/month12/temperature {{-40.0}};"
greenland.h5
```

The difference in this simpler command line is the default datatype of H5T_NATIVE_FLOAT and dataspace of SCALAR are used since a datatype and a dataspace are not specified. The DATA keyword was again left out.

The quotation marks enclosing the CREATE command are used so that the command line interface will interpret the parameters correctly.

1.2.2. The COPY Command and Parameters

The COPY command can be used to create an exact copy of an existing attribute as another attribute of the same or different target-object. If the new attribute is of the same target-object, it must use a different name. When the COPY command is used, the following is the syntax:

```
COPY <old_attribute_name> <new_attribute_name>;
```

The syntax elements are described in the table below.

Element	Comments
COPY	The command name. Upper case only.
<old_attribute_name>	The complete name of the existing attribute. For more information, see the "Attribute Names" section on page 13.
<new_attribute_name>	The name of the new attribute. If the new attribute will be an attribute of the same object as the existing attribute, then only the value of this parameter should be the name of the attribute. If the new attribute will be an attribute of a different object, then the value

Element	Comments
	of this parameter should be the complete name of the attribute (in other words, include the link names).
;	A terminator. Semicolons are used at the end of each command and parameter set.

The following are some examples.

```
COPY /group1/dataset1/attribute1 /group1/dataset2/attribute2_new;
```

With the command above, a copy of attribute1 of /group1/dataset1 is created with the name of attribute2_new of /group1/dataset2. Both attributes exist as two separate attributes.

```
COPY /group1/dataset1/attribute1 attribute1_new;
```

With the command above, a copy of attribute1 of /group1/dataset1 is created with the name of attribute1_new of the same /group1/dataset1. Both attributes exist as two separate attributes of the same dataset.

```
COPY /group1/dataset1/attribute1 /group1/dataset1/attribute1_new;
```

The command above produces the same effect as the previous example.

1.2.3. The DELETE Command and Parameter

The `DELETE` command can be used to delete an existing attribute from a specified object in an HDF5 data file. When the `DELETE` command is used, the following is the syntax:

```
DELETE <old_attribute_name>;
```

The syntax elements are described in the table below.

Element	Comments
`DELETE`	The command name. Upper case only.
`<old_attribute_name>`	The complete name of the attribute. For more information, see the “Attribute Names” section on page 13.
;	A terminator. Semicolons are used at the end of each command and parameter set.

The following is an example.

```
DELETE /All_Data/VIIRS-M1-SDR_All/Radiance/MeasurementUnits;
```

With the command above, `h5edit` will delete the `MeasurementUnits` attribute that is attached to the `VIIRS-M1-SDR_ALL` object.

1.2.4. The MODIFY Command and Parameters

The MODIFY command and parameters can be used to change the value(s) of an existing attribute. When the MODIFY command is used, the following is the syntax:

```
MODIFY <attribute_name> <attribute_data> ;
```

The syntax elements are described in the table below.

Element	Comments
MODIFY	The command name. Upper case only.
<attribute_name>	The complete name of the attribute. For more information, see the "Attribute Names" section on page 13.
<attribute_data>	Describes the new data value of the attribute. For more information, see the "Attribute Definitions" section on page 6.
;	A terminator. Semicolons are used at the end of each command and parameter set.

The following is an example.

```
MODIFY /All_Data/VIIRS-M1-SDR_All/Radiance/MaxValue {150.0};
```

With the command above, the data value of the attribute MaxValue will be 150.0. Note that the data value is enclosed in curly brackets. Note also that this command only changes data values: the datatype and dataspace of the attribute are not changed. For more information, see the "Attribute Definitions" section on page 14.

1.2.5. The RENAME Command and Parameters

The RENAME command can be used to change the name of an existing attribute. When the RENAME command is used, the following is the syntax:

```
RENAME <old_attribute_name> <new_attribute_name>;
```

The syntax elements are described in the table below.

Element	Comments
RENAME	The command name. Upper case only.
<old_attribute_name>	The complete name of the existing attribute. For more information, see the "Attribute Names" section on page 13.
<new_attribute_name>	The new name of the attribute. Since the attribute will still be an attribute of the same object, the value of this parameter should be the name of the attribute: the complete name of the attribute is not

Element	Comments
	needed.
;	A terminator. Semicolons are used at the end of each command and parameter set.

The following is an example.

```
RENAME /group1/dataset1/attribute1 attribute2;
```

With this command, `attribute1` of `/group1/dataset1` becomes `attribute2` of `/group1/dataset1`. The attribute named `attribute1` no longer exists.

1.3. Atomicity Levels

Use the `--atomic` command-line option to specify the atomicity of the operation. With this option, users can customize how changes are written to disk.

In the command syntaxes shown above, the atomicity level is specified with the following command-line option and parameter:

```
--atomic atomic_level
```

where

`--atomic` is the command-line option, and

`atomic_level` is the parameter. The atomic levels are described in the table below and are not case-sensitive.

Level	Comments
Yes	With this parameter, all of the changes are applied if the operation is successful, or none of the changes are applied if there is an error.
No	With this parameter, a backup copy of the HDF5 file will not be made, and the operations will be done as much as possible. A file might be partially modified when the no parameter is used.
Inc	Inc is short for incremental. This parameter might be used when there are several commands in a command line. After each command has been completed successfully, the changes are written (flushed) to disk. For example, if there were three commands on the command line, there would be three flushes if all of the operations were successful. This is the default level.

1.4. Data Recovery

When `h5edit` successfully opens an HDF5 file, it makes a backup copy of the file. If the operations are successful, `h5edit` will remove the backup file, but if the operations are unsuccessful, `h5edit` will leave the backup copy of the file for the user to replace the original file.

The name of the backup copy of the HDF5 file will have a file name that is based on the file name of the HDF5 file. The differences are that the backup file name will begin with a period and will end with the .bck extension. For example, if the data file name is “2010_10_01_data.h5”, the backup file name will be “.2010_10_01_data.h5.bck”.

1.5. Using an h5edit Command File

Users might prefer using the `--command-file` command-line option rather than the `--command` command-line option when running `h5edit`. With a command file, more complex command lines can be developed and stored for re-use. The other advantage is command-line shell meta-character issues can be ignored.

The command-line syntax that uses the `--command-file` command-line option is below.

```
h5edit --command-file command_file_name --atomic atomic_level
hdf5_file_name
```

The `--command-file` command-line option has one parameter, `command_file_name`. Command files are plain text files that hold `h5edit` command statements. See the “Commands and Command Parameters” section on page 6 for more information.

The following shows two features of using the `--command` command-line option: a long, complicated command line that creates two attributes and some shell meta-characters.

```
$ h5edit --command ”CREATE /sta1/month12/ScalarString {DATATYPE
{H5T_STRING {STRSIZE 15}} DATASPACE {SCALAR} DATA {\”scalar string\”}};
CREATE /sta1/month12/ArrayString {DATATYPE { H5T_STRING {STRSIZE 10}}
DATASPACE {SIMPLE ( 3 )} DATA {\”an\”, \”array\”, \”string\”}}; ”
greenland.h5
```

Note that the double quotes inside of the command string must be escaped so that the Unix shell will not treat them as the closing quotes.

The result of the example above can be achieved using a command file. The example below is the command-line syntax; the command file is called `strings_attributes`.

```
$ h5edit --command-file strings_attributes greenland.h5
```

The content of the command file is below.

```
CREATE /sta1/month12/ScalarString
   {DATATYPE {H5T_STRING {STRSIZE 15}}
    DATASPACE {SCALAR}
    DATA {”scalar string”}
   };
CREATE /sta1/month12/ArrayString
   {DATATYPE {H5T_STRING {STRSIZE 10}}
    DATASPACE {SIMPLE ( 3 )}
    DATA {”an”, ”array”, ”string”}
   };
```

The command file is easier to read because it does not need the various escape characters.

2. Reference

2.1. Attribute Names

In HDF5, objects such as datasets and groups are not named. The names applied to objects come from the names of the links that an application will use to get from the root group to the object. The set of links from the root group to an object is also known as the link path.

Attributes do have names. Attributes are attached to objects. Since more than one attribute may be attached to an object, attributes are given names so that they can be distinguished from each other.

While any ASCII or UTF-8 character may be used in the name given to an attribute, it is usually wise to avoid the following kinds of characters:

- Commonly used separators or delimiters such as slash, backslash, colon, and semicolon (\, /, :, ;)
- Escape characters
- Wild card characters such as asterisk and question mark (*, ?)

NULL can be used within a name, but HDF5 names are terminated with a NULL: whatever comes after the NULL will be ignored by HDF5.

In the command-line parameters in this document, attribute names and link paths are used to specify attributes. Depending on the operation, link paths may not be needed. An example of an attribute name is `MaxValue`. An example of an attribute name and the link path to the object that the attribute is attached to is `/All_Data/VIIRS-M1-SDR_All/Radiance/MaxValue`.

The general form of a complete attribute name is the following:

```
/link_name[/link_name…]/attribute_name
```

The first time an attribute is specified on the `h5edit` command line, the name of the attribute and its link path should be used. This tells `h5edit` where to look for the attribute. With some operations such as `RENAME`, a second attribute name needs to be specified on the command line. With `RENAME`, the second attribute name can be just the attribute name since the location (the object to which the attribute is attached) is already known. See the example below.

```
RENAME /All_Data/VIIRS-M1-SDR_All/Radiance/MaxValue MaxValue_New;
```

With `COPY`, the second attribute name may also need a link path if the new attribute will be attached to a different object. See the example below.

```
COPY /All_Data/VIIRS-M1-SDR_All/Radiance/MeasurementUnits
/All_Data/VIIRS-M1-SDR_All/Radiance_Updated/MeasurementUnits;
```

An alternate way of showing an attribute name and its link path is to separate the attribute name from the link path. The general form is the following:

```
/link_name[/link_name…] attribute_name
```

The RENAME example above would be written as the following:

```
RENAME /All_Data/VIIRS-M1-SDR_All/Radiance MaxValue MaxValue_New;
```

See the "HDF5 Attributes" chapter in the *HDF5 User's Guide* for more information.

2.2. Attribute Definitions

When an attribute is defined using the CREATE command, the following syntax must be used:

```
{DATATYPE DATATYPE_NAME {DATATYPE_PARAMETER} DATASPACE DATASPACE_TYPE
DATA {data_value}}
```

The entire definition must be enclosed in curly brackets.

Keywords must use upper case letters. In the syntax above, `DATATYPE`, `DATASPACE`, and `DATA` are constant keywords. `DATATYPE_NAME`, `DATATYPE_PARAMETER`, and `DATASPACE_TYPE` are variables for which the valid values are keywords. See the "Supported Datatypes" section on page 15 and the "Supported Dataspaces" section on page 17 for more information. Valid values for `data_value` are not keywords.

The `DATATYPE_PARAMETER` and `data_value` variables are enclosed in curly brackets. Valid values for `DATATYPE_PARAMETER` might include an integer if the datatype is a string and the size of the string is specified. Valid values for `data_value` will probably not be a keyword. There is no need to enclose valid values for `DATATYPE_NAME` and `DATASPACE_TYPE` with curly brackets: the valid values are keywords. See the "Supported Datatypes" section on page 15 and the "Supported Dataspaces" section on page 17 for more information.

If `DATATYPE` and `DATATYPE_NAME` are not included, the default datatype `H5T_FLOAT_NATIVE` will be used.

If `DATASPACE` and `DATASPACE_TYPE` are not included, the default dataspace `SCALAR` will be used.

An example is `{DATATYPE H5T_NATIVE_FLOAT DATASPACE SCALAR DATA {100.00}}`. The example starts with the keyword `DATATYPE` and `H5T_NATIVE_FLOAT`, the `DATATYPE_NAME`. Some datatypes may have a parameter. A string type may have a string size. If a `DATATYPE_PARAMETER` is used, it should be enclosed in curly brackets. See the "Supported Datatypes" section on page 15 for more information. The attribute's dataspace is defined with the `DATASPACE` keyword and `SCALAR`, the `DATASPACE_TYPE`. See the "Supported Dataspaces" section on page 17 for more information. The final part of the definition is the keyword `DATA` and `{100.00}`, the `data_value`. The data value should be enclosed in curly brackets.

The DATATYPE, DATASPACE, and DATA keywords are optional. The example in the paragraph above is {DATATYPE H5T_NATIVE_FLOAT DATASPACE SCALAR DATA {100.00}}. It could also be entered as {H5T_NATIVE_FLOAT SCALAR {100.00}}. The datatype and dataspace names are keywords recognized by h5edit, and data values are enclosed in curly brackets.

2.3. Attribute Data

The DATA keyword specifies the value that will be put into a newly created attribute. Use the following syntax:

```
DATA {data_value}
```

DATA is a constant keyword and is case-sensitive.

data_value will be some value and will be enclosed with curly brackets.

data_value must match the datatype and dataspace. For example, a data value of 100.00 with its decimal point would match a floating point datatype such as H5T_FLOAT_NATIVE, but a data value of 100.00 would not match an integer datatype such as H5T_STD_I8BE.

2.4. Supported Datatypes

An attribute may have one of the datatypes listed below. If a datatype is not specified, the default value, H5T_NATIVE_FLOAT, will be used. There are three kinds of supported datatypes: integer, floating point, and string.

2.4.1. Integer Datatypes

Keyword	Datatype Characteristics
H5T_STD_I8BE	Signed big endian 8 bit integer
H5T_STD_I16BE	Signed big endian 16 bit integer
H5T_STD_I32BE	Signed big endian 32 bit integer
H5T_STD_I64BE	Signed big endian 64 bit integer
H5T_STD_I8LE	Signed little endian 8 bit integer
H5T_STD_I16LE	Signed little endian 16 bit integer
H5T_STD_I32LE	Signed little endian 32 bit integer
H5T_STD_I64LE	Signed little endian 64 bit integer
H5T_STD_U8BE	Unsigned big endian 8 bit integer

Keyword	Datatype Characteristics
H5T_STD_U16BE	Unsigned big endian 16 bit integer
H5T_STD_U32BE	Unsigned big endian 32 bit integer
H5T_STD_U64BE	Unsigned big endian 64 bit integer
H5T_STD_U8LE	Unsigned little endian 8 bit integer
H5T_STD_U16LE	Unsigned little endian 16 bit integer
H5T_STD_U32LE	Unsigned little endian 32 bit integer
H5T_STD_U64LE	Unsigned little endian 64 bit integer

The following integer types are of the C programming language and are machine dependent. They are indicated by the following keywords:

Keyword	Datatype Characteristics
H5T_NATIVE_CHAR	Native char
H5T_NATIVE_UCHAR	Native unsigned char
H5T_NATIVE_SHORT	Native signed short
H5T_NATIVE_INT	Native signed int
H5T_NATIVE_LONG	Native signed long
H5T_NATIVE_LLONG	Native signed long long
H5T_NATIVE_USHORT	Native unsigned short
H5T_NATIVE_UINT	Native unsigned int
H5T_NATIVE_ULONG	Native unsigned long
H5T_NATIVE_ULLONG	Native unsigned long long

2.4.2. Floating Point Datatypes

The following floating point types are supported and are indicated by the corresponding keywords:

Keyword	Datatype Characteristics
H5T_IEEE_F32BE	IEEE big endian 32 bit floating point
H5T_IEEE_F64BE	IEEE big endian 64 bit floating point
H5T_IEEE_F32LE	IEEE little endian 32 bit floating point
H5T_IEEE_F64LE	IEEE little endian 64 bit floating point

The following floating point types are of the C programming language and are machine dependent. They are indicated by the following keywords:

Keyword	Datatype Characteristics
H5T_NATIVE_FLOAT	Native float
H5T_NATIVE_DOUBLE	Native double
H5T_NATIVE_LDOUBLE	Native long double

2.4.3. String Datatypes

The string type is supported and is identified by the keyword H5T_STRING. The string type has two parameters: the size of the string and the padding mechanism.

The string size is indicated by the keyword STRSIZE followed by a positive integer value of the string size.

The padding mechanism is indicated by the keyword STRPAD and the keyword H5T_STR_NULLTERM. H5T_STR_NULLTERM, null terminated as in the C programming language, is currently the only padding mechanism.

Note that this version of h5edit supports only fixed size strings and null terminated padding.

2.5. Supported Dataspaces

An attribute may have one of the dataspaces listed below. If a dataspace is not specified, the default value, SCALAR, will be used.

Keyword	Dataspace Characteristics
SCALAR	The attribute has a single data element.
SIMPLE	The SIMPLE dimension dataspace means the attribute's dataspace is a multi-dimensional array. The dataspace is identified by the SIMPLE keyword and is followed by the dimension sizes enclosed in a pair of parenthesis. The rank of the dimension is deduced from the number of dimension sizes specified. For example, SIMPLE (2,3,4) is a 3-dimensional array of 24 data elements total.

The HDF5 Library supports other dataspaces, but these other dataspaces such as the NULL dataspace are not yet supported by h5edit. Note also that unlimited dimension sizes are not yet supported.

2.6. Keywords

A number of keywords have been identified. These are reserved for use by the HDF5 Library and h5edit. The keywords are listed below in alphabetical order. Keywords should always be entered with upper case letters.

```
COPY
CREATE
DATA
DATASPACE
DATATYPE
DELETE
H5T_IEEE_F32BE
```

```
H5T_IEEE_F32LE
H5T_IEEE_F64BE
H5T_IEEE_F64LE
H5T_NATIVE_CHAR
H5T_NATIVE_DOUBLE
H5T_NATIVE_FLOAT
H5T_NATIVE_INT
H5T_NATIVE_LDOUBLE
H5T_NATIVE_LLONG
H5T_NATIVE_LONG
H5T_NATIVE_SHORT
H5T_NATIVE_UCHAR
H5T_NATIVE_UINT
H5T_NATIVE_ULLONG
H5T_NATIVE_ULONG
H5T_NATIVE_USHORT
H5T_STD_I16BE
H5T_STD_I16LE
H5T_STD_I32BE
H5T_STD_I32LE
H5T_STD_I64BE
H5T_STD_I64LE
H5T_STD_I8BE
H5T_STD_I8LE
H5T_STD_U16BE
H5T_STD_U16LE
H5T_STD_U32BE
H5T_STD_U32LE
H5T_STD_U64BE
H5T_STD_U64LE
H5T_STD_U8BE
H5T_STD_U8LE
H5T_STR_NULLPAD
H5T_STR_NULLTERM
H5T_STR_SPACEPAD
H5T_STRING
MODIFY
NULL
RENAME
SCALAR
SIMPLE
STRPAD
STRSIZE
```

3. Examples

In the examples in this chapter, the command values `CREATE`, `COPY`, `DELETE`, `MODIFY`, and `RENAME` are in bold only to make reading the examples in the document easier.

3.1. Add Attributes to a File

The command-line statement below adds four attributes to the file `SVM01_ter_ grav_dev.h5`. The first two are unsigned short (2 bytes) attributes. The third one is a string type attribute. The last one is a floating point attribute. Note that the backslash indicates a line continuation for the Unix shell. This is needed by some Unix shells such as C shell.

```
$ h5edit -c " \
CREATE /All_Data/VIIRS-M1-SDR_All/Radiance/FillValue-SOUB_UINT16_FILL
{DATATYPE H5T_STD_U16LE DATASPACE SCALAR DATA {65528}} ; \
CREATE /All_Data/VIIRS-M1-SDR_All/Radiance/FillValue-NA_UINT16_FILL {DATATYPE
H5T_STD_U16LE DATASPACE SCALAR DATA {65535}} ; \
CREATE /All_Data/VIIRS-M1-SDR_All/Radiance/MeasurementUnits {DATATYPE
H5T_STRING {STRSIZE 6 } DATASPACE SCALAR DATA { \"W/m^2\" }} ; \
CREATE /All_Data/VIIRS-M1-SDR_All/Radiance/MaxValue {DATATYPE
H5T_NATIVE_FLOAT DATASPACE SCALAR DATA {100.00}} ; \
" \
SVM01_ter_ grav_dev.h5
```

3.1.1. Adding Attributes Using a Command File

The command file version of the command-line statement above is shown below.

```
$ h5edit –command-file add_attr SVM01_ter_ grav_dev.h5
```

The command file `add_attr` is shown below.

```
CREATE /All_Data/VIIRS-M1-SDR_All/Radiance/FillValue-SOUB_UINT16_FILL
    {DATATYPE H5T_STD_U16LE DATASPACE SCALAR DATA {65528}};
CREATE /All_Data/VIIRS-M1-SDR_All/Radiance/FillValue-NA_UINT16_FILL
    {DATATYPE H5T_STD_U16LE DATASPACE SCALAR DATA {65535}};
CREATE /All_Data/VIIRS-M1-SDR_All/Radiance/MeasurementUnits
    {DATATYPE H5T_STRING {STRSIZE 6} DATASPACE SCALAR DATA { \"W/m^2\" }};
CREATE /All_Data/VIIRS-M1-SDR_All/Radiance/MaxValue
    {DATATYPE H5T_NATIVE_FLOAT DATASPACE SCALAR DATA {100.00}};
```

As the example above shows, one advantage to using a command file is there is no need to worry about meta-characters that are interpreted by a Unix shell.

3.2. Delete Attributes from a File

The command-line statement below deletes all of the attributes created by the command above in the "Add Attributes to a File" section. The backslash is again used to indicate a line continuation for the Unix shell. This is needed by some Unix shells such as C shell.

```
$ h5edit -c " \
DELETE /All_Data/VIIRS-M1-SDR_All/Radiance/FillValue-SOUB_UINT16_FILL; \
DELETE /All_Data/VIIRS-M1-SDR_All/Radiance/FillValue-NA_UINT16_FILL; \
DELETE /All_Data/VIIRS-M1-SDR_All/Radiance/MeasurementUnits; \
DELETE /All_Data/VIIRS-M1-SDR_All/Radiance/MaxValue; \" \
SVM01_ter_ grav_dev.h5
```

3.2.1. Deleting Attributes Using a Command File

The command file version of the command-line statement in the "Delete Attributes from a File" section above is shown below.

```
$ h5edit –command-file delete_attr SVM01_ter_ grav_dev.h5
```

The command file `delete_attr` is shown below.

```
DELETE /All_Data/VIIRS-M1-SDR_All/Radiance/FillValue-SOUB_UINT16_FILL;
DELETE /All_Data/VIIRS-M1-SDR_All/Radiance/FillValue-NA_UINT16_FILL;
DELETE /All_Data/VIIRS-M1-SDR_All/Radiance/MeasurementUnits;
DELETE /All_Data/VIIRS-M1-SDR_All/Radiance/MaxValue;
```

As the example above shows, one advantage to using a command file is there is no need to worry about meta-characters that are interpreted by a Unix shell.

3.3. Copy Attributes from One Object to Another

```
COPY /group1/dataset1/attribute1 /group1/dataset2/attribute2_new;
```

With the command above, a copy of `attribute1` of `/group1/dataset1` is created with the name of `attribute2_new` of `/group1/dataset2`. Both attributes exist as two separate attributes.

```
COPY /group1/dataset1/attribute1 attribute1_new;
```

With the command above, a copy of `attribute1` of `/group1/dataset1` is created with the name of `attribute1_new` of the same `/group1/dataset1`. Both attributes exist as two separate attributes of the same dataset.

```
COPY /group1/dataset1/attribute1 /group1/dataset1/attribute1_new;
```

The command above produces the same effect as the previous example.

3.4. Modify Attributes

```
MODIFY /All_Data/VIIRS-M1-SDR_All/Radiance/MeasurementUnits {'W/m/m'};
```

In the command above, the value of attribute `MeasurementUnits` is changed to "W/m/m".

```
MODIFY /All_Data/VIIRS-M1-SDR_All/Radiance MaxValue {150.0};
```

In the command above, the value of attribute `MaxValue` is increased to 150.0.

```
MODIFY /All_Data/VIIRS-M1-SDR_All/Radiance/MaxValue {150.0};
```

The command above produces the same effect as the previous example.

```
MODIFY /All_Data/VIIRS-M1-SDR_All/Radiance MaxValue {150};
```

The command above is illegal because the attribute is of the float datatype but the data is of the integer type. A decimal point would indicate this is a floating point value.

3.5. Rename Attributes

```
RENAME /group1/dataset1/attribute1 attribute2;
```

With the command above, `attribute1` of `/group1/dataset1` becomes `attribute2` of `/group1/dataset1`, and `attribute1` ceases to exist.

```
RENAME /group1/dataset1/attribute1 /group1/dataset1/attribute2;
```

This is illegal because the `<new_attribute_name> attribute2` should be only the attribute name and not the full path name.