RFC: Supporting soft-link and external-link for h5diff

Jonathan Kim (jkm@hdfgroup.org)

This RFC is for discussing about a new feature for supporting soft-link and external-link when ‘follw link’ command option is given for h5diff as our Chicago customer requested for the feature in h5diff command line tool.

# Description

Currently, h5diff command only compares the name (path) of link value not the actual target object when dealing with link(s).

With the new feature, h5diff will go through the link(s) and figure out the object at the end of the link and compare its data value and attribute if exist.

Internally there are 3 kinds of links, which are hard-link, soft-link, and external-link (as part of user defined-link). The new feature will provide transparent experience to a user when comparing linked objects as the user can specify any of those links to be compared in any combination.

# Test results without following link option as example

## Test cases were created to demonstrate current h5diff behavior. There are 3 HDF5 data files were used. One file is for soft-link tests and the other two files are for external-link tests. The output from h5dump for each file is shown below.

* Soft-link test file contents:

HDF5 "soft\_link.h5" {

GROUP "/" {

 SOFTLINK "softlink\_dset1\_1" {

 LINKTARGET "target\_dset1"

 }

 SOFTLINK "softlink\_dset1\_2" {

 LINKTARGET "target\_dset1"

 }

 SOFTLINK "softlink\_dset2" {

 LINKTARGET "target\_dset2"

 }

 SOFTLINK "softlink\_group1" {

 LINKTARGET "target\_group"

 }

 SOFTLINK "softlink\_group2" {

 LINKTARGET "target\_group"

 }

 SOFTLINK "softlink\_noexist" {

 LINKTARGET "no\_obj"

 }

 DATASET "target\_dset1" {

 DATATYPE H5T\_STD\_I32LE

 DATASPACE SIMPLE { ( 2, 4 ) / ( 2, 4 ) }

 DATA {

 (0,0): 0, 1, 2, 3,

 (1,0): 1, 2, 3, 4

 }

 }

 DATASET "target\_dset2" {

 DATATYPE H5T\_STD\_I32LE

 DATASPACE SIMPLE { ( 2, 4 ) / ( 2, 4 ) }

 DATA {

 (0,0): 0, 0, 0, 0,

 (1,0): 0, 0, 0, 0

 }

 }

 GROUP "target\_group" {

 DATASET "dset" {

 DATATYPE H5T\_STD\_I32LE

 DATASPACE SIMPLE { ( 2, 4 ) / ( 2, 4 ) }

 DATA {

 (0,0): 0, 1, 2, 3,

 (1,0): 1, 2, 3, 4

 }

 }

 }

}

}

* External-link source file contents:

HDF5 "extlink\_source.h5" {

GROUP "/" {

 EXTERNAL\_LINK "ext\_link\_dset1" {

 TARGETFILE "extlink\_target.h5"

 TARGETPATH "/target\_group/x\_dset"

 DATASET "/target\_group/x\_dset" {

 DATATYPE H5T\_STD\_I32LE

 DATASPACE SIMPLE { ( 2, 4 ) / ( 2, 4 ) }

 DATA {

 (0,0): 0, 1, 2, 3,

 (1,0): 1, 2, 3, 4

 }

 }

 }

 EXTERNAL\_LINK "ext\_link\_dset2" {

 TARGETFILE "extlink\_target.h5"

 TARGETPATH "/target\_group2/x\_dset"

 DATASET "/target\_group2/x\_dset" {

 DATATYPE H5T\_STD\_I32LE

 DATASPACE SIMPLE { ( 2, 4 ) / ( 2, 4 ) }

 DATA {

 (0,0): 0, 0, 0, 0,

 (1,0): 0, 0, 0, 0

 }

 }

 }

 EXTERNAL\_LINK "ext\_link\_grp1" {

 TARGETFILE "extlink\_target.h5"

 TARGETPATH "target\_group"

 GROUP "target\_group" {

 DATASET "x\_dset" {

 HARDLINK "/target\_group/x\_dset"

 }

 }

 }

 EXTERNAL\_LINK "ext\_link\_grp2" {

 TARGETFILE "extlink\_target.h5"

 TARGETPATH "target\_group2"

 GROUP "target\_group2" {

 DATASET "x\_dset" {

 HARDLINK "/target\_group2/x\_dset"

 }

 }

 }

 EXTERNAL\_LINK "ext\_link\_noexist1" {

 TARGETFILE "extlink\_target.h5"

 TARGETPATH "no\_obj"

 }

 EXTERNAL\_LINK "ext\_link\_noexist2" {

 TARGETFILE "no\_file.h5"

 TARGETPATH "no\_obj"

 }

}

}

* External-link target file contents:

HDF5 "extlink\_target.h5" {

GROUP "/" {

 DATASET "target\_dset1" {

 DATATYPE H5T\_STD\_I32LE

 DATASPACE SIMPLE { ( 2, 4 ) / ( 2, 4 ) }

 DATA {

 (0,0): 0, 1, 2, 3,

 (1,0): 1, 2, 3, 4

 }

 }

 GROUP "target\_group" {

 DATASET "x\_dset" {

 DATATYPE H5T\_STD\_I32LE

 DATASPACE SIMPLE { ( 2, 4 ) / ( 2, 4 ) }

 DATA {

 (0,0): 0, 1, 2, 3,

 (1,0): 1, 2, 3, 4

 }

 }

 }

 GROUP "target\_group2" {

 DATASET "x\_dset" {

 DATATYPE H5T\_STD\_I32LE

 DATASPACE SIMPLE { ( 2, 4 ) / ( 2, 4 ) }

 DATA {

 (0,0): 0, 0, 0, 0,

 (1,0): 0, 0, 0, 0

 }

 }

 }

}

}

## Test results

* Soft-link test results:

===============================================================

# Case: group (soft linked) vs group (target)

CMD> h5diff -v soft\_link.h5 soft\_link.h5 /softlink\_group1 /target\_group

</softlink\_group1> is of type H5G\_LINK and </target\_group> is of type H5G\_GROUP

--------------------------------

Some objects are not comparable

--------------------------------

Use -c for a list of objects.

===============================================================

# Case: dset (soft linked) vs dset (target)

CMD> h5diff -v soft\_link.h5 soft\_link.h5 /softlink\_dset1\_1 /target\_dset1

</softlink\_dset1\_1> is of type H5G\_LINK and </target\_dset1> is of type H5G\_DATASET

--------------------------------

Some objects are not comparable

--------------------------------

Use -c for a list of objects.

===============================================================

# Case: dset (soft linked) vs non-exist object

CMD> h5diff -v soft\_link.h5 soft\_link.h5 /softlink\_dset1\_1 /softlink\_noexist

link : </softlink\_dset1\_1> and </softlink\_noexist>

1 differences found

===============================================================

# Case: group (soft linked and exist) vs group (soft linked and exist)

CMD> h5diff -v soft\_link.h5 soft\_link.h5 /softlink\_group1 /softlink\_group2

link : </softlink\_group1> and </softlink\_group2>

0 differences found

===============================================================

# Case: dset (soft linked and exist) vs dset (soft linked and exist) - same values

CMD> h5diff -v soft\_link.h5 soft\_link.h5 /softlink\_dset1\_1 /softlink\_dset1\_2

link : </softlink\_dset1\_1> and </softlink\_dset1\_2>

0 differences found

===============================================================

# Case: dset (soft linked and exist) vs dset (soft linked and exist) - diff values

CMD> h5diff -v soft\_link.h5 soft\_link.h5 /softlink\_dset1\_1 /softlink\_dset2

link : </softlink\_dset1\_1> and </softlink\_dset2>

1 differences found

* External-link test results

===============================================================

# Case group (ext linked in src file) vs group (in target file)

CMD> h5diff -v extlink\_source.h5 extlink\_target.h5 /ext\_link\_grp1 /target\_group

</ext\_link\_grp1> is of type H5G\_UDLINK and </target\_group> is of type H5G\_GROUP

--------------------------------

Some objects are not comparable

--------------------------------

Use -c for a list of objects.

===============================================================

# Case dset (ext linked in src file) vs dset (in target file)

CMD> h5diff -v extlink\_source.h5 extlink\_target.h5 /ext\_link\_dset1 /target\_group/x\_dset

</ext\_link\_dset1> is of type H5G\_UDLINK and </target\_group/x\_dset> is of type H5G\_DATASET

--------------------------------

Some objects are not comparable

--------------------------------

Use -c for a list of objects.

===============================================================

# Case dset (ext linked) vs object (non-exist / exsiting file)

CMD> h5diff -v extlink\_source.h5 extlink\_target.h5 /ext\_link\_dset1 /ext\_link\_noexist1

Object </ext\_link\_noexist1> could not be found in <extlink\_target.h5>

===============================================================

# Case dset (ext linked) vs object (non-exist / no file)

CMD> h5diff -v extlink\_source.h5 extlink\_target.h5 /ext\_link\_dset1 /ext\_link\_noexist2

Object </ext\_link\_noexist2> could not be found in <extlink\_target.h5>

===============================================================

# Case group (ext linked and exist) vs group (ext linked and exist)

CMD> h5diff -v extlink\_source.h5 extlink\_source.h5 /ext\_link\_grp1 /ext\_link\_grp2

external link: </ext\_link\_grp1> and </ext\_link\_grp2>

1 differences found

===============================================================

# Case dset (ext linked and exist) vs dset (ext linked and exist)

CMD> h5diff -v extlink\_source.h5 extlink\_source.h5 /ext\_link\_dset1 /ext\_link\_dset2

external link: </ext\_link\_dset1> and </ext\_link\_dset2>

1 differences found

# Decision from suggestions for following link behavior

**We decided to go with the ‘suggestion 2’ at the discussion with the RFC1.0.**

**For the reference, all the previous suggestions recorded here.**

Two simple cases of using h5diff

* For comparing the two files : h5diff <file1> <file2>
* For comparing the two object : h5diff <file1> <file2> <obj1> <job2>

## Suggestion 1

Act as the current h5diff behavior, which perform diff only if both <obj1> and <obj1> is same type. So follow link only if the both <obj1> and <obj2> are same type as link.

* File compare case - If ‘follow option’ is given, h5diff will follow links since it only compares when the both objects are same type.
* Object compare case – If ‘follow option’ is given, check if both are link type, if so, continue and will follow links to compare. If not, it’s ‘Not compatible’

## Suggestion 2

Follow links like the diff command does in unix – go through link if any of given objects are either soft-link or external-link. It will perform diff either both given objects are links (soft/external/hard) or one of objects is link (soft/external/hard) as well as none of the given objects is link (only hard).

* If ‘follow option’ is given, h5diff will follow links to figure out the end-target object and compares when the both objects are same type. If the type is dataset, h5diff will compare all the values and attribute. If the type is group, h5diff will compare the name and attribute. If the type is named-type h5diff will compare the name and attribute.

## Suggestion 3

Give user control via ‘follow option’ with below arguments

- both – follow link if both <obj1> and <obj2> are link , otherwise it’s ‘Not compatible’

- src – follow <obj1> if it’s link, otherwise it’s ‘Not compatible’

- dst – follow <obj2> if it’s link, otherwise it’s ‘Not compatible’

* File compare case – if ‘follow option’ is given,
	+ both- follow link if both types are link , otherwise it’s ‘Not compatible’
	+ src – follow links in file1 , not file2. This will always be ‘Not compatible’
	+ dst – follow links in file2, not file1. This will always be ‘Not compatible’
* Object compare case – if ‘follow options’ is given,
	+ both – follow links if both are link
	+ src – follow obj1, if it’s link , otherwise it’s ‘Not compatible’
	+ dst – follow obj2, if it’s link, otherwise it’s ‘Not compatible’

# Command option for comparing through links

**For the option, we ‘-f’ or ‘-l’ were considered. After some investigation, ‘-l’ or ‘—link-follow’ options will be used to compare through links. This is based on ‘h5dump’ also used –l for supporting link. Other unix command or h5tools doesn’t use any specific options for link.**

* Unix ln command
	+ –s for symbolic link
	+ Create hard links by default, symbolic links with --symbolic.
	+ -f is used for force, flag,
* H5ls
	+ -f : print full path name
	+ –s : print 1-byte integer
	+ –l : label members of compound datasets
* H5copy
	+ –f : flag with args (shallow, soft, ext, ref, noattr…)
	+ –s : source obj name
	+ No –l
* H5dump
	+ –f D : --filedriver=D : Specify which driver to open the file with
	+ **-l P, --soft-link=P Print the value(s) of the specified soft link**
	+ No -s

## Option example

* h5diff **–l** <h5\_src\_file> <h5\_dst\_file> /link\_to\_dset1 /link\_to\_dset2
	+ This will perform diff since both source and target type is link.
* h5diff **–l**  <h5\_src\_file> <h5\_dst\_file> /link\_to\_dset1 /dset2
	+ This will perform diff since one of the specified object is link..
* h5diff **–l** <h5\_src\_file> <h5\_dst\_file> /dset1 /dset2

Without these options, the behavior would stay same as now.

Acknowledgements

The Chicago customer requested for the feature in h5diff command line tool. Thanks Peter for guideline.

Revision History

|  |  |
| --- | --- |
| *December 18, 2009:* | Version 1 circulated for comment within The HDF Group.  |
| *Jan 10, 2009:* | Version 1.1 circulated for comment within The HDF Group |