URL: http://cxc.harvard.edu/ciao3.4/merging_rules.html
Last modified: December 2006

AHELP for CIAO 3.4

merging_rules

Context: concept

Jump to: Description See Also

Synopsis

A description of the merging rules used when combining header information.

Description

Merging rules have been implemented in the header library and are mainly used by the dmmerge tool. The purpose of this document is to describe what each of the rules means, and also give an example for each rule.

Rules with examples

(1) MAX

Take the maximal value among the keys that are present in the input files. If the key is not found in any of the input files, then no output and warning for the key.

Example:

- infile1: KEY = 4.
- infile2: KEY = 2.5
- infile3: KEY = 3.
- outfile: KEY = 4.

[In this case, the output is the maximal value '4'.]

(2) MIN

Take the minimal value among the keys that are present in the input files. If the key is not found in any of the input files, then no output and warning for the key.

Example:

- infile1: KEY = 4.
- infile2: KEY = 2.5
- infile3: KEY = 3.
- outfile: KEY = 2.5

[In this case, the output is the minimal value '2.5'.]

(3) FAIL

If the key is not present in the first file, or it is present in the first file but the value is different from others, then display a warning and don't write the key to the output. If the key is present in the first file and the value is same as others, then write the value to the output file without a warning.

Example:

- infile1: KEY = 4.
- infile2: KEY = 2.5
- infile3: KEY = 3.
- outfile: (KEY not present)

[In this case, KEY is present in the first file, but the value is different from others. Therefore display a warning: 'KEY values are different...FAIL...', but no output for KEY.]

(4) FAIL;Default−[value]

First, go through each input file. If the key is not present, set the key value to the default (i.e. [value]). Once each file is checked, compare the values with one another. If they are all same, write it to the output; otherwise, display a warning but no output for the key.

Example using the rule 'FAIL;Default−3':

- infile1: (KEY not present)
- infile2: (KEY not present)
- infile3: KEY = 3
- outfile: KEY = 3

[In this case, KEY is not present in the infile1 and infile2, so the rule sets both of them to be the default value '3', which happens to be same as the one in infile3. Therefore, it writes '3' to the output without a warning.]

(5) calcForce

Calculate the value with a special hard−coded rule and write it to the output file. Currently, hard−coded rules are available only for the keys "DATE", "TSTART", and "TSTOP". For other keys, this rule will simply copy the first value to the output file.

Notes:

The following are special keys whose merging rules are always treated like 'calcForce' in the dmmerge tool. In other words, they are always computed from the special hard−coded rules:

DATE−OBS DATE−END ONTIME LIVETIME EXPOSURE

(6) FORCE

First, go through each input file. If the key is not present, set the key value to the default (null, 0, blank). Once each file is checked, compare the values with one another. If they are different, write the first one to the output with a warning message; otherwise, output the value without warning.

Example:

- infile1: KEY = 3
- infile2: (KEY not present)
- infile3: KEY = 5
- outfile: KEY = 3

[In this case, the KEY of infile2 is not present, therefore set it to the default (i.e. '0') first. Since KEY values are different, write the first one (i.e. '3') to the output with a warning: 'KEY has 2 different values.']

(7) FORCE−[value]

First, go through each input file. If the key is not present, set the value to the default (i.e. [value]). Once each file is checked, compare the values with one another. If they are different, write the first one to the output with a warning message; otherwise, output the value without warning.

Example using the rule 'Force−3':

- infile1: KEY = 3
- infile2: (KEY not present)
- infile3: KEY = 3
- outfile: KEY = 3

[In this case, the KEY of infile2 is not present, therefore set it to [value] (i.e. '3') first. Since KEY values are all same, write it to the output without warning.]

(8) WarnFirst

If the key is not present in the first file, give a warning and don't write it to the output. Otherwise, copy the first value to the output and display a warning message only if the first value is different from others.

Example:

- infile1: (KEY not present)
- infile2: KEY = 5
- infile3: KEY = 5
- outfile: (KEY not present)

[In this case, the KEY value of infile1 is not present, therefore no output for KEY except a warning message: "omit − keyword KEY not present in 1st file"]

(9) WarnFirst;Force−[value]

First, go through each input file. If the key is not present, set the value to the default (i.e. [value]). After each

file is checked, copy the first value to the output; if it is different from others, display a warning message.

Example using the rule 'WarnFirst;Force−5':

- infile1: (KEY not present)
- infile2: KEY = 5
- infile3: KEY = 5
- outfile: KEY = 5

[In this case, the KEY value of infile1 is not present, therefore force it to [value] (i.e. '5') first. Since all KEYs have same value, write it to the output without a warning.]

(10) WarnOmit−[tol]

If the key is not present in the first header, or if it is present in the first header but different from others more than [tol], then don't write the key and give a warning. Otherwise, take the first value.

Example using the rule 'WarnOmit−0.3':

- infile1: KEY = 5.5
- infile2: KEY = 5
- infile3: KEY = 5
- outfile: (KEY not present)

[In this case, the KEY value of infile1 is present but is different from others more than the given tolerance (i.e. 0.3), therefore no output for KEY except a warning: 'omit − KEY values different more than 0.3'.]

(11) WarnPrefer−[value]

If the key is not present in the first file, or if it is present in the first file but different from others, then write the [value] to the output and display a warning. If all the key values are same, then write it to the output with no warning.

Example using the rule 'WarnPrefer−9.0':

- infile1: (KEY not present)
- infile2: KEY = 5
- infile3: KEY = 5
- outfile: KEY = 9.0

[In this case, KEY is not present in infile1, therefore set it to [value] (i.e. '9.0'). Since the KEY values are different, write the first one to the output with a warning: 'warning: KEY has 2 different values'.]

(12) Merge−[value1];Force−[value2]

First, go through each input file. If the key is not present, force it to [value2]. Once each file is checked, compare the values with one another. If different, then write [value1] to the output with a warning. If same, write the first value to the output.

Example using the rule 'MERGE−99;FORCE−10':

- infile1: (KEY not present)
- infile2: KEY = 5
- infile3: KEY = 5
- outfile: KEY = 99

[In this case, the KEY is not present in infile1, therefore set it to [value2] (i.e. '10') first. Since the KEY values are different, write [value1] (i.e. '99') to the output and display a warning: 'KEY has different value..merged..'.]

(13) SKIP

Don't write the key to the output file.

Example:

- infile1: KEY = 5
- infile2: KEY = 5
- infile3: KEY = 5
- outfile: (KEY not present.)

[In this case, always omit KEY from the output file.]

(14) PUT_STRING−[value]

Force the output value to be [value]. It applies for the key that does not belong to the event header and has the data type of 'char'.

Example using the rule 'PUT_STRING−TEST':

- infile1: KEY = 'test1'
- infile2: KEY = 'test2'
- infile3: KEY = 'test3'
- outfile: KEY = 'TEST'

[In this case, force the KEY value to be 'TEST'.]

(15) PUT_DOUBLE−[value]

Force the output value to be [value]. It applies for the key that does not belong to the event header and has the data type of 'DOUBLE'.

Example using the rule 'PUT_DOUBLE−99.9':

- infile1: KEY = 100.0
- infile2: KEY = 200.0
- infile3: (KEY not present)

- outfile: KEY = 99.9

[In this case, force the KEY value to be 99.9.]

(16) PUT_LONG−[value]

Force the output value to be [value]. It applies for the key that does not belong to the event header and has the data type of 'LONG'.

Example using the rule 'PUT_LONG−100':

- infile1: KEY = 99
- infile2: KEY = 99
- infile3: KEY = 99
- outfile: KEY = 100

[In this case, force the KEY value to be 100.]

Special merging rules and keywords

(1) All merging rules described in the above section can only be applied for the keys that belong to the event header except the following rules of 1.a, 1.b, and 1.c.

1.a. These three rules should be used by the keys that do not belong to the event header:

- PUT_STRING−[value]
- PUT_DOUBLE−[value]
- PUT_LONG−[value]

1.b. This rule can be used by any key, regardless whether or not it belongs to the event header:

- SKIP

1.c. The above description for this rule is only applied to an event key. For a non−event key, it will be same as the rule 'PUT_STRING−[value]':

- FORCE−[value]

(2) The merging rules of the following keys are always treated like 'calcForce' in the dmmerge tool. In other words, they are always computed from the special hard−coded rules:

DATE−OBS DATE−END ONTIME LIVETIME EXPOSURE

About the lookup table

A lookup table is required by the dmmerge tool. It assigns a merging rule to each keyword.

(1) Example of lookup table

MISSION	Merge−Merged;Force−AXAF
TELESCOP	Merge−Merged;Force−Unknown
OBJECT	Merge−Merged;Force−Unknown
OBSERVER	Merge−Merged;Force−Unknown
TITLE	Merge−Merged;Force−Unknown
OBS_ID	Merge−Merged;Force−Unknown
INSTRUME	Merge−Merged;Force−Unknown
DETNAM	Merge−Merged;Force−Unknown
GRATING	Merge−Merged;Force−Unknown
OBS_MODE	Merge−Merged;Force−Unknown
DATAMODE	Merge−Merged;Force−Unknown
ORIGIN	WarnFirst;Force−ASC
CONTENT	Force
HDUNAME	Force
RA_NOM	WarnOmit−0.0003
DEC_NOM	WarnOmit−0.0003
ROLL_NOM	WarnOmit−1.0
SIM_X	WarnOmit−0.001
SIM_Y	WarnOmit−0.001
SIM_Z	WarnOmit−0.001
FOC_LEN	WarnOmit−1.0
EQUINOX	WarnPrefer−2000.0
RADECSYS	WarnPrefer−ICRS
DATACLAS	WarnPrefer−Observed
TIMVERSN	WarnFirst
CLOCKAPP	WarnFirst
MJDREF	fail
TIERRELA	Max
TIERABSO	max
TIMEDEL	maX
DATE	calcForce
DATE−OBS	calcForce
DATE−END	calcForce
TSTART	calcForce
TSTOP	calcForce
ONTIME	calcForce
LIVETIME	calcForce
DTCOR	calcForce
TIMEUNIT	Fail;Default−s
TIMESYS	Fail;Default−UTC

TIMEPIXR	Fail;Default−0.5

(2) Lookup Table Format

o) Each row consists of two columns: the keyword and the associated merging rule.

o) The two columns are separated by blanks or tabs.

o) If a default value is assigned to a merging rule, use "−" to separate them.

e.g. mergeRule−defValue

where defValue can be a number or a character string

o) If a keyword has two merging rules together, use ";" as a delimiter between the merging rules.

For example:

mergeRuleOne;mergingRuleTwo

mergeRuleOne−defValueOne;mergeRuleTwo−defValueTwo

(3) Lookup Table Restriction

For the first column:

o) It has to be in capitals.

For the second column:

o) It has to be one word without any space

o) If a default value is assigned to a merging rule, its data type needs to be same as the keyword. Otherwise, a warning message will be displayed.

o) Currently not able to have a negative number as the numerical default value.

o) For those keywords not found in the user's lookup table, the code will copy the 1st header to the output.

o) Merging rules are limited to those defined in Sec. 3.

o) For any merging rule not found in Sec. 3, the 1st value will be copied to the output.

o) Merging rules are only applied to the values of the keyword, not to the 'comment' of the keyword.

See Also

tools

dmimgcalc, dmmerge

URL: http://cxc.harvard.edu/ciao3.4/merging_rules.html
Last modified: December 2006