dpssvali

August 16, 2005

Abstract

The task **dpssvali** performs validation checks of XMM PPS files making it the automatic part of the Data Product Screening System (DPSS). The tests include existence and parameter range checks for header key words and source tables columns of EPIC PPS Fits files. The output of the validation is written to the files 'invalid.log' and 'valid.log'. The checks to be carried out are coded in ascii files, which can be easily modified. Each ascii check file contains a sequence of commands which refer to a certain group of XMM PPS products, e.g. bkgmap.chk for all EPIC background maps. The task **dpssvali** selects a certain check file for execution in dependence of the PPS product type.

1 Instruments/Modes

Instrument	Mode
EPIC	IMAGING, TIMING, BURST
OM	

2 Use

pipeline processing	yes
interactive analysis	no

3 Description

The SAS task **dpssvali** is part of the Data Product Screening System (DPSS), which is divided into an automatic step (described here) and an subsequent visual verification step based on Graphical User Interfaces. The separation into an automatic validation and a visual verification process has been proven as an effective and powerful method to secure a high quality of data products produced by a standard analysis software. The SAS task **dpssvali** is written in `perl` and depends on the SAS task **dscheck**. For a certain group of XMM PPS products (e.g. EPIC background maps, EPIC maximum-likelihood source lists, EPIC images, EPIC sensitivity maps) a list of commands is executed stored in an ascii file (hereafter check file). Each ascii check file contains a sequence of commands which refer to a certain group of XMM PPS products, e.g. bkgmap.chk for all EPIC background maps. The task **dpssvali** selects a certain check file for execution in dependence of the PPS product type. The tests include existence and parameter range checks for header key words and source tables columns of EPIC PPS Fits files. The output of the

validation is written to the files 'invalid.log' and 'valid.log'. The checks to be carried are coded in ascii files, which can be easily modified.

The following ascii check files are presently available:

- bkgmap.chk

  This check file performs existence and parameter range checks for EPIC PN, MOS1, MOS2 background maps in the energy bands 0,1,2,3,4 and 5. The existence checks include the header keywords `CONTENT`, `INSTRUME`, `DATAMODE`, `OBS_ID`, `EXP_ID`, `DATE-OBS`, `DATE-END`. Parameter range checks are performed for the header keywords `RA_PNT` (check fails, if `RA_PNT` is outside (0,360)), `DEC_PNT` (-90,90), `EQUINOX` (1950,2000), `EXPOSURE` (5000,75600).

- expmap.chk

  Existence and parameter range checks for EPIC PN, MOS1, MOS2 exposure maps in the energy bands 1,2,3,4,5,7.

- exsnmp.chk

  Existence and parameter range checks for EPIC PN, MOS1, MOS2 sensitivity maps in the energy bands 1,2,3,4,5,7.

- omsrli.chk

  This check file performs existence and parameter range checks as described above (for the `omsrli.chk` file) for EPIC Maximum-Likelihood Source lists. In addition, the secondary header content is validated. The check fails, if the source list SRCLIST does not exists. The following source list table columns are expected to be present (the allowed parameter range is given in brackets): `SRC_INST` (0,3), `ID_BAND` (0,5), `SCTS` (1,100000), `FLUX` (0,1e-4), `RA` (0,360), `DEC` (-90,90), `HR1` (-1,1), `HR2` (-1,1), `HR3` (-1,1)

- image.chk

  This check file performs existence and parameter range checks for EPIC PN, MOS1, MOS2 and OM images (except OM sky images) in the energy bands 1,2,3,4,5,7 (naming convention `TTTTTT = IMAGE_`). The performed checks are identical to the checks coded in the `bkgmap.chk` file. In addition the check file performs existence checks for the presence of the World Coordinate System (WCS) in the primary header.

- imageom.chk

  This check file performs existence and parameter range checks for OM sky images `TTTTTT = SIMAGE`). The performed checks are identical to the checks coded in the `image.chk` file, except the range check for the exposure, which should now be within the range of 500 to 75600 seconds.

- swsrli.chk

  The check file performs eixstence and parameter range checkes for OM detect source lists for different exposures (`TTTTTT = SWSRLI`). Primary header existence and parameter range checkes as for other OM PPS products mentioned above, except the range check for the magnitude `MAG` which should be within 5 and 30.

- obsrls.chk

  The check file performs eixstence and parameter range checkes for OM detect lists for different exposures (`TTTTTT = OBSRLS`). Primary header existence and parameter range checkes as for other OM PPS products mentioned above, except the range check for the U magnitude `MAG_U` which should be within 5 and 30.

- mod8mp.chk

  This check file performs existence and parameter range checks for OM MOD MAP images `TTTTTT = MOD8MP`. The performed checks are identical to the checks coded in the `imageom.chk` file.

- flafld.chk

  This check file performs existence and parameter range checks for OM FLAT FIELD images `TTTTTT = FLAFLD`. The performed checks are identical to the checks coded in the `imageom.chk` file.

4 Parameters

This section documents the parameters recognized by this task (if any).

Parameter	Mand	Type	Default	Constraints
set	yes	string	ppsid.fit	only EPIC and OM files names are allowed

PPS file name

checkdir	no	directory	.	

pathname to check files

validlog	no	file	valid.log	

name of the output DPSS file containing PPS files which passed without any error the task

invalidlog	no	file	invalid.log	

name of the output DPSS file containing PPS files which did not pass the task

5 Errors

This section documents warnings and errors generated by this task (if any). Note that warnings and errors can also be generated in the SAS infrastructure libraries, in which case they would not be documented here. Refer to the index of all errors and warnings available in the HTML version of the SAS documentation.

ARG_INPUT *(error)*
Dataset not readable

WRGNAM *(error)*
Filename does not comply to naming convention

NOCHKS *(error)*
Can't determine checks for file

ARG_CHKDIR *(error)*
Directory `checkdir` for tests not found

MISCHECK *(error)*
Checkfile `chkfile` not readable or missing

OPVAL" *(error)*
Can't open `validlog` file

OPIVAL" *(error)*
Can't open `invalidlog` file

FAIL *(warning)*
Header keyword and/or table column entry not existing or out of limits
corrective action: Check pipeline processing

6 Input Files

1. EPIC PPS products according to naming convention given in [1] and SSC-LUX-TN-0038.
   (a) EPIC Background Maps: TTTTTT = BKGMAP
   (b) EPIC and OM Images: TTTTTT = IMAGE_
   (c) EPIC Maximum-Likelihood Detection Source List: TTTTTT = EMSRLI
   (d) EPIC Exposure Maps: TTTTTT = EXPMAP
   (e) EPIC Sensitivity Maps: TTTTTT = EXSNMP
   (f) OM Sky Images: TTTTTT = SIMAGE
   (g) OM Detect List: TTTTTT = OMSRLI
   (h) OM Source List: TTTTTT = SWSRLI
   (i) OM Flat Field Image: TTTTTT = FLAFLD
   (j) OM Mod Map Image: TTTTTT = MOD8MP

7 Output Files

1. valid.log

   This ascii file contains a list of XMM PPS products names which passed the SAS task **dpssvali**.

2. invalid.log

   This file contains XMM PPS product names which were found to be invalid, i.e. which did not pass the SAS task **dpssvali**.

8 Algorithm

The SAS task dpssvali is written in perl.

```
**** define parameters
my $fitsfile="";
my $chkdir   = "";
my $validlog = "";
my $invalidlog = "";



**** start of subroutine dpssvali
sub dpssvali{
$fitsfile=stringParameter("set");
$chkdir=stringParameter("checkdir");
$validlog=stringParameter("validlog");
$invalidlog=stringParameter("invalidlog");


**** check file permission
unless (-r $fitsfile) {

    SAS::error("ARG_INPUT", "Dataset '$fitsfile' not readable");
}

****  check existence of ascii check files in given directory
unless (-d $chkdir) {
    SAS::error("ARG_CHKDIR", "Directory '$chkdir' for tests not found");
}

***   open DPSS output files validlog and invalidlog

open VLOG, ">> $validlog" or
 SAS::error("OPVAL", "Can't open validlog '$validlog': $!");
open ILOG, ">> $invalidlog" or
 SAS::error("OPIVAL", "Can't open invalidlog '$invalidlog': $!");

  print "in=$fitsfile\nDir=$chkdir\nval=$validlog\ninval=$invalidlog\n";



*** check existence of a check file attributed to a PPS product file name

my @chkfiles = determine_checks($fitsfile);
unless (@chkfiles) {
    SAS::error("NOCHKS", "Can't determine checks for file: '$fitsfile'\n");
}

*** list check files which resulted into invalid PPS products

my @failed = check_fitsfile($fitsfile, @chkfiles);
if (@failed) {
```

```
   SAS::error("FAIL", "File '$fitsfile' failed the following tests: " .
join(', ', @failed) . "\n");
} else {
   exit 0;
}
}

*** perform checks according a selected PPS product file name

sub determine_checks {
    my $file = shift;

    $file =~ s|.*/||;
    my $t6;
    # here real algorithm for checkfiles ... use dummys now
    if ( $file =~ /^P\d{10}\w\w\d{3}(\w{6})\d{5}\.\w{3}$/) {
$t6 = lc($1);
    } else {
        SAS::error("WRGNAM", "Filename does not comply to naming convention: '$file'\n");
    }

    my @chks = ();
    @chks = $chkdir . '/expmap.chk'   if $t6 eq 'expmap';
    @chks = $chkdir . '/bkgmap.chk'   if $t6 eq 'bkgmap';
    @chks = $chkdir . '/exsnmp.chk'   if $t6 eq 'exsnmp';
    @chks = $chkdir . '/image.chk'    if $t6 =~ /image_|oimage/;
    @chks = $chkdir . '/mod8mp.chk'   if $t6 eq 'mod8mp';
    @chks = $chkdir . '/flafld.chk'   if $t6 eq 'flafld';
    @chks = $chkdir . '/imageom.chk'  if $t6 eq 'simage';
    @chks = $chkdir . '/omsrli.chk'   if $t6 eq 'omsrli';
    @chks = $chkdir . '/obsrls.chk'   if $t6 eq 'obsrls';
    @chks = $chkdir . '/swsrli.chk'   if $t6 eq 'swsrli';

    @chks;
}


**** list of XMM PPS files which failed the tests

sub check_fitsfile {
    my $file = shift;
    my (@checks) = @_;

    my @failed_tests;
    foreach my $chkfile ( @checks ) {
       unless ( -r $chkfile ) {
            SAS::error("MISCHECK", "Checkfile '$chkfile' not readable or missing\n");
       }
       system "dscheck --set=$file --checkfile=$chkfile"
    and push @failed_tests, $chkfile;
    }

    if (@failed_tests) {
       print ILOG "$fitsfile: ", join(', ', @failed_tests), "\n";
    } else {
```

```
      print VLOG "$fitsfile", "\n";
    }
    return @failed_tests
}
```

9 Comments

- RGS files are not yet included. **dpssvali** uses **dscheck** created by Richard West.

10 Future developments

Including RGS data sets. Adding additional functionality to EPIC PPS product checks according to input from SAS developers.

References

[1] SSC. XMM Survey Science Centre to Science Operations ICD for SSC Products. Technical Report XMM-SOC-ICD-0006-SSC Issue 2.1, SSC, Mar 2000.