EUMETSAT POLAR SYSTEM

Kai Users Guide

EUMETSAT

Am Kavalleriesand 31, Postfach D-64295 Darmstadt, Germany

Tel: +49 6151.8077 * Fax: +49 6151.807.553

EUMETSAT POLAR SYSTEM	EPS Programme: Kai Users Guide	EUMETSAT Ref.:EUM.EPS.TEN.04.014 Issue: 2 Revision 1 WBS Number: 240000 Date: 19 January 2011

© 2011, EUMETSAT

DOCUMENT CHANGE LOG

Issue	Revision	Date	DCR no.	Pages Affected	Reason for change
1	1	21 January 2005	-	-	First creation of the document released along with v1.1 of the software.
2	1	19 January 2011			Updated program help text.

1 INTRODUCTION

1.1 Purpose and Scope

This document describes the Kutter Assembler and Integrator tool *kai*, covering installation, usage and a description of the algorithms used.

1.2 Document Structure

Section 1 – Introduction.

Section 2 – Overview of the capabilities of the tool.

Section 3 – Installation guidelines.

Section 4 – Instructions for using the software.

Section 5 – Overall description of the processing steps used.

Section 6 – Detailed description of how the MPHR record is treated.

1.3 Referenced documents

RD.1 EPS GPFS, EPS.GGS.SPE.96167, Issue 6 Revision 4

RD.2 EPFS_XML, EPS.SYS.TEN.04.001, Issue 1 Revision 0

RD.3 Eugene Users Guide, EPS.SYS.TEN.02.030, Issue 2 Revision 0

RD.4 libxml2 homepage, www.xmlsoft.org

RD.5 doxygen homepage, www.stack.nl/~dimitri/doxygen

1.4 Applicable documents

AD.1 Kai web site http://tctrac/projects/kai (EUMETSAT internal network only)

2 OVERVIEW

kai is a tool for processing EPS products [RD.1]. It has 6 main functions:

- To split a large product into one or more smaller valid products, for example to create 3-minute PDUs from a full orbit product.

- To combine smaller products together into a large product.

- To repair existing products by correcting bad fields in the MPHR, rebuilding the IPR records and setting timestamps of the SPHR, GIADR and GEADR records.

- To build complete EPS products out of records stored individually in files. The separate files will be ordered and concatenated together and if needed the MPHR and IPR records will be synthesized.

- As a simple viewer to display the MPHR, SPHR, IPR, GEADR, VEADR and VIADR-L0-OBT2UTC records, and of the generic record headers for all other records. This function is similar to the `quickread` tool supplied with Eugene but with some additional information in the output text including the names of records (instead of just the numerical RECORD_SUBCLASS) and full content of all the record types defined in the GPFS.

- Split an EPS product and place each record into a different file.

Kai can be used with all EPS products. It uses EPFS_XML files [RD.2] to identify record names and to set two fields in the MPHR (FORMAT_MAJOR_VERSION and FORMAT_MINOR_VERSION). It is safe to use the tool with any products of any version because in general the records are treated as 'black boxes' and not modified.

3 INSTALLATION

3.1.1 Prerequisites

- A POSIX-compliant system (AIX, Solaris, Linux, cygwin are known to work).
- C++ compiler.
- libxml2 [RD.4]. Version 2.5.4 has been tested but other versions should work.
- (optional) The Eugene tool from http://tctrac/projects/eugene (EUMETSAT internal network) or available from http://www.eumetsat.int externally.

The version numbers given above are the versions used during development and kai may also work with older versions of the tools and libraries.

3.2 Compilation

Kai uses the GNU autoconf build system. To install it a user should typically run:

```
tar zxvf kai-x.y.tar.gz
cd kai-x.y
./configure
make
make install
```

To target an install directory other than the default (/usr/local) pass the flag '—prefix <<dir>>' to the ./configure command..

3.3 Additional documentation

Using the `doxygen` [RD.5] program a detailed design document can be generated by running `make doxygen`. This will be in HTML format and also PDF provided `pdflatex` is installed.

The files created are put into the `doxygen` directory.

4 USAGE

4.1 kai

4.1.1 Overview

Perform record-level operations on input EPS products.

4.1.2 Inputs

One of more EPS full products, or records stored in individual files.

4.1.3 Outputs

EPS products, records stored in individual files or text display.

4.1.4 Usage

usage: kai `[OPTIONS]`

where `OPTIONS` can include:

`[-i | --in] <input file(s)`	Specify one or more input files. The `-i` or `--in` flags are optional. To use the quickread text display mode, do not specify any output flags (`--out`, `--pdu` or `-split`). Only one input file can be given in quickread mode, otherwise multiple inputs may be given.
`-o, --out <output file (prefix)>`	Specify the output file name. This is not used if the `--pdu` or `--split` options are chosen.

`--starttime <generalised time> | +<number> <ms|s|m>`	Specify start time of first MDR to be used in any outputs, either as a UTC time or as an offset in milliseconds (ms), seconds (s) or minutes (m) from start time of the first MDR in the inputs.
	Ie.
	`--starttime 20020808183000Z`
	`--starttime +102m`
	`--starttime +500000ms`
	The `--starttime` option is applied to all output types, including a single product, PDUs, split records or text display.

	If omitted the earliest start time found in the input products will be used.
`--endtime <generalised time> | +<number> <ms|m|s>`	Specify start time of the last MDR to be included, applies to all output types. If omitted the timestamp of the last MDR found in the input files will be used.
`-e, --extract SPEC`	Extract only the given record types from the input file(s) and to use in the output file(s). No modifications will be applied (ie. the normal IPR and MPHR processing is suppressed). This means the output file(s) will may not be valid EPS products if there is no MPHR extracted. SPEC is a colon separated list of conditions which will e tested against input records. Each condition can either be: 'class=<class>' where <class> is a class name ('mdr', 'ipr' etc) or a RECORD_CLASS number. 'subclass=<number>' 'instrument=<insturment>' where <instrument is an instrument name of INSTRUMENT_GROUP number. 'name=<name>' to match records by name, where <name> is a record name defined in the format's EPFS_XML file. 'range=<range>' to specify a numberical range of records from an array, ie. '1,5,10-20', or '5-' to select records from number 5 onwards or '-5' to select records up to an including number 5. If the `--split` option is used the output will be one record per file, otherwise all extracted records will be concatenated into a single file.
`-r, --remove SPEC`	All records matching SPEC (see definition above under `--extract`) are eliminated from the output file(s).
`-c, --count <number>`	Include only `number` of MDRs in output file(s). The MDRs are simply counted out in the order they appear in the input file(s) without regard to the subclass of the MDRs.
`-s, --skip <number>`	Skip `number` of MDRs before starting processing
`-p, --pdu, --pdus <filename prefix>`	Split input product into PDUs. The files will be named <filename-prefix>.1.pdu, <filename-prefix>.2.pdu … This cannot be combined with the `--out` or `--split` options.
`--pdutime [<duration><ms|s|m>]`	Override the default PDU length of 3 minutes to the

	supplied value.
`--split <filename prefix>`	Split product into 1 file per record. The name given is a prefix, files are given a full name describing their record type, a counter and a .dat file extension. This cannot be combined with the `--out` or `--pdu` options.
`--subsat <filename>`	Set subsatellite positions filename. See Section 6 for description of how this is used. If not specified then all files matching EUGENE_HOME/share/subsat-*.txt are scanned. See Section 4.1.5 for derivation of EUGENE_HOME.
`--osvs <filename>`	Set orbit state vector file. If not specified then EUGENE_HOME/share/osvs.xml is assumed.
`--epfsxml <dirname>`	Set directory to read EPFS_XML files from. If not specified then EUGENE_HOME/formats is assumed.
`--slice <file>`	Split product into slices.
`--statefile <file>`	Location of state file
`--slice-duration <n>`	Main duration of slices; default=180s
`--slice-pre-time <n>`	Lead-in overlap time for slices; default=94s
`--slice-post-time <n>`	Lead-out overlap time for slices
`-h, --help`	Show help page.
`-v, --version`	Show version number and build date.

Run ‘kai –help’ to see options and examples online.

4.1.5 Environment variables

kai needs to know the location of the Eugene data files for some functionality. A default directory is set at compile time in the make.config file, which can be overridden by setting the $EUGENE_HOME environment variable at runtime. Run `kai -help` to check what the current value being used is. This should point to the location where the Eugene tarball file was extracted to, and does not need to contain a compiled executable.

4.1.6 Output file names

Kai produces output files when the ‘—output’, ‘—pdu’ or ‘—slice’ options are used. Each

of these options takes a single string as a parameter.
If this string is a directory name the output file or files will be written to that location and named according to the GPFS rules.
Otherwise, the string will be taken as the literal name to use (if a single output is generated with the —output option) or as a filename prefix (with the –pdu or –slice options).
The tool eps_renamer can be used to rename files to their GPFS-compliant names if needed.

4.2 Quickread

4.2.1 Overview

Quickread is a simple EPS product viewer, showing in text format the content of record headers, ASCII records and other information from the GPFS document (IPR and viadr-l0-obt2utc records are fully decoded).

4.2.2 Inputs

An EPS product

4.2.3 Outputs

Text to terminal

4.2.4 Usage

Usage is identical to Kai except that no output options can be specified. Normal usage is just ‘quickread <filename>’.

The output is automatically piped through a pager if output is being written to a terminal.

4.2.5 Environment variables

Quickread uses 2 additional environment variables:
$PAGER : Used to specify an alternate pager to less.

$NO_COLOUR : If set to any value the output will not include any colour codes.

5 PROCESSING STEPS

To process a set of input products and produce output products kai goes through the following steps:

5.1 Record sorting and duplicate removal

Each input file is scanned for records, which are placed into a long array in the order specified in [RD.1]. As each record is inserted there is a check for duplicate VIADR and VEADR records, which are discarded. This check is implemented as a binary comparison over the full record, and is done so that a set of PDUs can be recombined back into a product within spurious records.
In addition there is a simpler check for duplicate MPHR, SPHR, GIADR and GEADR records – only the first record found in each of these classes, or subclasses for GIADR and GEADR records, is placed into the array.

Any IPR and DMDR records in the input files are discarded at this point.

5.2 Filtering

If the –extract, --remove, --count, –skip, --start-time or –end-time options are in use, the total records read are filtered and any that the user did not require are discarded.

5.3 Split into output files

At this step the read records are split into one or more output files depending on flags passed.

5.4 IPR and DMDR generation

In the next step, for each output list (either per PDU, or for a single product) a set of IPR records is generated. DMDR records are added as needed, using a CCSDS counter based algorithm for L0 products and a time-based algorithm for L1 products.

5.5 MPHR generation

If no MPHR records were read in a blank MPHR is created at this point. It will have all of its fields set in the next step.

5.6 MPHR editing

Each MPHR field is modified, as described in detail in Section 6.

This process will usually produce a fully valid MPHR consistent with the rest of the product, however it will not work for L0 products if the MPHR had to be built from scratch.

6 MPHR FIELDS

List the MPHR fields & how each one is modified/set/not touched and displayed.

PRODUCT_NAME

This is set (after all other MPHR fields) according to [RD.1].

PARENT_PRODUCT_NAME_1

Not modified.

PARENT_PRODUCT_NAME_2

Not modified.

PARENT_PRODUCT_NAME_3

Not modified.

PARENT_PRODUCT_NAME_4

Not modified.

INSTRUMENT_ID

Set to the fist 4 letters of the instrument name, right padded with 'x' up to 4 characters if needed.

INSTRUMENT_MODEL

Set to ' 1' if it was previously either blank or 'xxx'.

PRODUCT_TYPE

This is set to 'xxx' unless:

- The product is an ATOVS or IASI L2 product, then PRODUCT_TYPE is 'SND'
- ASCAT L1B products have product type set to SZF, SZO or SZR depending on the type of MDR present.
- IASI engineering and verification products have PRODUCT_TYPE set to ENG or VER.

PROCESSING_LEVEL

Set to 00, 01, 1A, 1B or 02 depending on the records found in the product.

SPACECRAFT_ID

Set to M01 if the input SPACECRAFT_ID was 'xxx' or blank.

SENSING_START

Set to the RECORD_START_TIME of the first MDR in the product.

SENSING_END

Set to the RECORD_STOP_TIME of the last MDR in the product.

SENSING_START_THEORETICAL

If the output is a new product being synthesized, see STATE_VECTOR_TIME below. Otherwise this field is set to the same values as the first parent product.

SENSING_END_THEORETICAL

If the output is a new product being synthesized, set to SENSING_START_THEORETICAL plus 102 minutes. Otherwise this field is set to the same values as the first parent product.

PROCESSING_CENTRE

Set to 'CGS1' if the input is blank or 'xxxx'.

PROCESSOR_MAJOR_VERSION

Set to '1' if the input is blank.

PROCESSOR_MINOR_VERSION

Set to '0' if the input is blank.

FORMAT_MAJOR_VERSION

The EPFS_XML format descriptions are needed to set the FORMAT_MAJOR/MINOR_VERSION fields. From these files a database is built with a mapping from each record type (class, subclass, version and instrument ID) to a text name and to a list of format versions. Since new PFS revisions do not have to redefine every record type, a single record can correspond to multiple MPHR format versions.

To set the FORMAT_*_VERSION fields the output product is scanned one record at a time until the algorithm finds a record for which only a single format version is defined, which is used to set the FORMAT_MAJOR_VERSION and FORMAT_MINOR_VERSION fields.

The format descriptions are found in EUGENE_HOME/formats, where EUGENE_HOME can be set at compile time or overridden using the environment variable $EUGENE_HOME.

FORMAT_MINOR_VERSION

See above.

PROCESSING_TIME_START

Set to SENSING_START plus 110 minutes if the input is set to a default value or to a time earlier that SENSING_START.

PROCESSING_TIME_END

Set to SENSING_START plus 115 minutes if the input is set to a default value or to a time earlier that SENSING_START.

PROCESSING_MODE

Set to 'N'.

DISPOSITION_MODE

Set to 'T'.

RECEIVING_GROUND_STATION

Set to ‘SVL’ if the input is blank or ‘xxx’.

RECEIVE_TIME_START

Set to SENSING_START plus 105 minutes.

RECEIVE_TIME_END

Set to SENSING_START plus 110 minutes.

ORBIT_START

See STATE_VECTOR_TIME.

If the orbit state vector match fails and ORBIT_START contains a non-zero value to begin with, it is not modified.

ORBIT_END

Set equal to ORBIT_START unless the product duration is equal to or greater than 102 minutes, in which case ORBIT_END is set to ORBIT_START plus 1.

ACTUAL_PRODUCT_SIZE

Set to the file size.

STATE_VECTOR_TIME

A list of orbit state vectors is read from EUGENE_HOME/osvs/osvs.xml. Each entry has a time giving the ascending node time and a list of parameters.

To set the orbit parameters of the MPHR the list of input OSVs is scanned to find the vector with the latest time stamp which is still less than SENSING_START. The time stamp of the vector used is copied to STATE_VECTOR_TIME and the parameters are copied into the next 12 fields.

If the time difference between SENSING_START and the OSV time is greater then 120 minutes the match fails and default values will be used for the orbit state vector, unless the MPHR already contains non-zero values in which case they will not be changed.

The field SENSING_START_THEORETICAL is set equal to STATE_VECTOR_TIME.

The OSV file also contains the orbit number which is written to ORBIT_START.

SEMI_MAJOR_AXIS

See above.

ECCENTRICITY

See above.

INCLINATION

See above.

PERIGEE_ARGUMENT

See above.

RIGHT_ASCENSION

See above.

MEAN_ANOMALY

See above.

X_POSITION

See above.

Y_POSITION

See above.

Z_POSITION

See above.

X_VELOCITY

See above.

Y_VELOCITY

See above.

Z_VELOCITY

See above.

EARTH_SUN_DISTANCE_RATIO

Not modified.

LOCATION_TOLERANCE_RADIAL

Not modified.

LOCATION_TOLERANCE_CROSSTRACK

Not modified.

LOCATION_TOLERANCE_ALONGTRACK

Not modified.

YAW_ERROR

Not modified.

ROLL_ERROR

Not modified.

PITCH_ERROR

Not modified.

SUBSAT_LATITUDE_START

A list of subsatellite positions is read from the files EUGENE_HOME/subsat/subsat-*.txt. Each entry in these files gives a date and a time in seconds, and a latitude/longitude pair.

The 2 SUBSAT_*_START fields are set equal to the subsatellite position at time SENSING_START.

If the lookup fails and the input product contains non-zero values in the MPHR, they are not changed.

SUBSAT_LONGITUDE_START

See above.

SUBSAT_LATITUDE_END

See above - the 2 SUBSAT_*_END fields are set equal to the subsatellite position at time SENSING_END.

SUBSAT_LONGITUDE_END

See above.

LEAP_SECOND

If LEAP_SECOND_UTC is set equal to the default time of midnight Jan 1 2000, leap second is set to '0'. Otherwise the field is not changed.

(nb. This is because some IASI products have LEAP_SECOND set incorrectly to –1 when in fact there is no leap second present in the product)

LEAP_SECOND_UTC

No changed.

TOTAL_RECORDS

Set to the total record count in the product.

TOTAL_MPHR

Set to the total MPHR record count in the product.

TOTAL_SPHR

Set to the total SPHR record count in the product.

TOTAL_IPR

Set to the total IPR record count in the product.

TOTAL_GEADR

Set to the total GEADR record count in the product.

TOTAL_GIADR

Set to the total GIADR record count in the product.

TOTAL_VEADR

Set to the total VEADR record count in the product.

TOTAL_VIADR

Set to the total VIADR record count in the product.

TOTAL_MDR

Set to the total MDR record count in the product.

COUNT_DEGRADED_INST_MDR

Not changed.

COUNT_DEGRADED_PROC_MDR

Not changed.

COUNT_DEGRADED_INST_MDR_BLOCKS

Not changed.

COUNT_DEGRADED_PROC_MDR_BLOCKS

Not changed.

DURATION_OF_PRODUCT

Set to the offset from SENSING_START to SENSING_END.

MILLISECONDS_OF_DATA_PRESENT

If the duration of the product has not been modified by kai (ie. a PDU split would change the duration) and the input MPHR contains a non-zero value in MILLISECONDS_OF_DATA_PRESENT it is not changed. Otherwise, it is set to the offset from SENSING_START to SENSING_END.

MILLISECONDS_OF_DATA_MISSING

Set to 0 unless the input MPHR contains a different value and the product duration has not been modified.

SUBSETTED_PRODUCT

Not changed.

APPENDIX A – SAMPLE USAGE

This section lists example uses of kai. All samples use start from an input ASCAT product named ASCA_SZF_1B_Z, although any EPS format product can be used.

Example 1

Display only the MDR headers of a product, showing the time difference in ms between adjacent records:

```
quickread --extract class=mdr --show-gaps ASCA_SZF_1B_Z
```

Example 2

Perform a repair operation on input product. The result will have the same data records as the input file, but the MPHR will have been processed to ensure all fields are consistent with the actual data, and any IPRs present in the input will have been removed and a new set created.

```
kai ASCA_SZF_1B_Z -o out
```

Example 3

Split a product into 3 minute PDUs named ascat-pdu.00000.pdu, ascat-pdu.00001.pdu etc.

```
kai ASCA_SZF_1B_Z --pdu ascat-pdu
```

Example 4

Remove all VEADR records of subclass 1 except the first 5, writing the result to a new file.

```
kai ASCA_SZF_1B_Z --remove class=veadr:subclass=1:range=5- -o my-
trimmedfile
```

Example 5

Assemble a group of PDUs into a single product, concatenated to only include 30 minutes of data.

```
kai ascat-pdu* --end-time +30m -o ascat-30mins
```

APPENDIX B – EXAMPLE OF PRODUCT DISPLAY

```
> kai ASCA-SZF-1B_Z


GENERIC_RECORD_HEADER for MPHR at offset 0 bytes:
   RECORD_CLASS 1 (MPHR)
   INSTRUMENT_GROUP 0 (GENERIC)
   RECORD_SUBCLASS 0 (mphr)
   RECORD_SUBCLASS_VERSION 2
   RECORD_SIZE 3307
   RECORD_START_TIME 950,65700719 (2002/08/08 18:15:00.719)
   RECORD_STOP_TIME 950,71805844 (2002/08/08 19:56:45.844)
   MPHR content:
   PRODUCT_NAME                  = GOME_xxx_00_M01_20020808181500Z_20020808195646Z_N_O_20020808195215Z
   PARENT_PRODUCT_NAME_1         = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
   PARENT_PRODUCT_NAME_2         = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
   PARENT_PRODUCT_NAME_3         = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
   PARENT_PRODUCT_NAME_4         = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
   INSTRUMENT_ID                 = GOME\n
   INSTRUMENT_MODEL              =   1\n
   PRODUCT_TYPE                  = xxx\n
   PROCESSING_LEVEL              = 00\n
   SPACECRAFT_ID                 = M01\n
   SENSING_START                 = 20020808181500Z\n
   SENSING_END                   = 20020808195646Z\n
   SENSING_START_THEORETICAL     = 20020808181317Z\n
   SENSING_END_THEORETICAL       = 20020808195646Z\n
   PROCESSING_CENTRE             = KSPT\n
   PROCESSOR_MAJOR_VERSION       =     1\n
   PROCESSOR_MINOR_VERSION       =     0\n
   FORMAT_MAJOR_VERSION          =     1\n
   FORMAT_MINOR_VERSION          =     0\n
   PROCESSING_TIME_START         = 20020808195215Z\n
   PROCESSING_TIME_END           = 20020808210854Z\n
   PROCESSING_MODE               = N\n
   DISPOSITION_MODE              = O\n
   RECEIVING_GROUND_STATION      = SVL\n
   RECEIVE_TIME_START            = 20020808195150Z\n
   RECEIVE_TIME_END              = 20020808195600Z\n
   ORBIT_START                   =     2\n
   ORBIT_END                     =     3\n
   ACTUAL_PRODUCT_SIZE           =   305500083\n
   STATE_VECTOR_TIME             = 20020808174340547Z\n
   SEMI_MAJOR_AXIS               =  7395171000\n
   ECCENTRICITY                  =       25814\n
   INCLINATION                   =      102728\n
   PERIGEE_ARGUMENT              =        1532\n
   RIGHT_ASCENSION               =       39389\n
   MEAN_ANOMALY                  =      358545\n
   X_POSITION                    =  5567915000\n
   Y_POSITION                    =  4571734000\n
   Z_POSITION                    =     -30000\n
   X_VELOCITY                    =     1049422\n
   Y_VELOCITY                    =    -1286033\n
   Z_VELOCITY                    =     7348501\n
   EARTH_SUN_DISTANCE_RATIO      =           0\n
   LOCATION_TOLERANCE_RADIAL     =           0\n
   LOCATION_TOLERANCE_CROSSTRACK =           0\n
   LOCATION_TOLERANCE_ALONGTRACK =           0\n
   YAW_ERROR                     =           0\n
   ROLL_ERROR                    =           0\n
   PITCH_ERROR                   =           0\n
   SUBSAT_LATITUDE_START         =       67066\n
   SUBSAT_LONGITUDE_START        =      232850\n
   SUBSAT_LATITUDE_END           =       65395\n
   SUBSAT_LONGITUDE_END          =      205725\n
   LEAP_SECOND                   =  0\n
   LEAP_SECOND_UTC               = xxxxxxxxxxxxxxx\n
   TOTAL_RECORDS                 =  16285\n
   TOTAL_MPHR                    =      1\n
   TOTAL_SPHR                    =      0\n
   TOTAL_IPR                     =      2\n
   TOTAL_GEADR                   =      0\n
   TOTAL_GIADR                   =      0\n
   TOTAL_VEADR                   =      0\n
```

```
   TOTAL_VIADR                   =      1\n
   TOTAL_MDR                     =  16281\n
   COUNT_DEGRADED_INST_MDR       =      0\n
   COUNT_DEGRADED_PROC_MDR       =      0\n
   COUNT_DEGRADED_INST_MDR_BLOCKS=      0\n
   COUNT_DEGRADED_PROC_MDR_BLOCKS=      0\n
   DURATION_OF_PRODUCT           =  6105000\n
   MILLISECONDS_OF_DATA_PRESENT  =  6092375\n
   MILLISECONDS_OF_DATA_MISSING  =        0\n
   SUBSETTED_PRODUCT             = F\n

GENERIC_RECORD_HEADER for IPR at offset 3307 bytes:
   RECORD_CLASS 3 (IPR)
   INSTRUMENT_GROUP 0 (GENERIC)
   RECORD_SUBCLASS 0 (IPR)
   RECORD_SUBCLASS_VERSION 1
   RECORD_SIZE 27
   RECORD_START_TIME 950,65700000 (2002/08/08 18:15:00.000)
   RECORD_STOP_TIME 950,71805000 (2002/08/08 19:56:45.000)
   IPR content:
   TARGET_RECORD_CLASS 7 (VIADR)
   TARGET_INSTRUMENT_GROUP 0 (GENERIC)
   TARGET_RECORD_SUBCLASS 0 (viadr-l0-obt2utc)
   TARGET_OFFSET 3361

GENERIC_RECORD_HEADER for IPR at offset 3334 bytes:
   RECORD_CLASS 3 (IPR)
   INSTRUMENT_GROUP 0 (GENERIC)
   RECORD_SUBCLASS 0 (IPR)
   RECORD_SUBCLASS_VERSION 1
   RECORD_SIZE 27
   RECORD_START_TIME 950,65700000 (2002/08/08 18:15:00.000)
   RECORD_STOP_TIME 950,71805000 (2002/08/08 19:56:45.000)
   IPR content:
   TARGET_RECORD_CLASS 8 (MDR)
   TARGET_INSTRUMENT_GROUP 0 (GENERIC)
   TARGET_RECORD_SUBCLASS 0 (mdr-l0)
   TARGET_OFFSET 4691

GENERIC_RECORD_HEADER for VIADR at offset 3361 bytes:
   RECORD_CLASS 7 (VIADR)
   INSTRUMENT_GROUP 0 (GENERIC)
   RECORD_SUBCLASS 0 (viadr-l0-obt2utc)
   RECORD_SUBCLASS_VERSION 2
   RECORD_SIZE 38
   RECORD_START_TIME 950,65700719 (2002/08/08 18:15:00.719)
   RECORD_STOP_TIME 950,71805844 (2002/08/08 19:56:45.844)
   viadr-l0-obt2utc content:
   UTC_OBT_TIME 2002/05/19 08:58:00.000 (869,32280000)
   OBT_UTC_TIME 0,32000
   CLOCK_STEP 3906249305

GENERIC_RECORD_HEADER for MDR at offset 3399 bytes:
   RECORD_CLASS 8 (MDR)
   INSTRUMENT_GROUP 0 (GENERIC)
   RECORD_SUBCLASS 0 (mdr-l0)
   RECORD_SUBCLASS_VERSION 1
   RECORD_SIZE 18764
   RECORD_START_TIME 950,65700719 (2002/08/08 18:15:00.719)
   RECORD_STOP_TIME 950,65700719 (2002/08/08 18:15:00.719)

GENERIC_RECORD_HEADER for MDR at offset 22163 bytes:
   RECORD_CLASS 8 (MDR)
   INSTRUMENT_GROUP 0 (GENERIC)
   RECORD_SUBCLASS 0 (mdr-l0)
   RECORD_SUBCLASS_VERSION 1
   RECORD_SIZE 18764
   RECORD_START_TIME 950,65701094 (2002/08/08 18:15:01.094)
   RECORD_STOP_TIME 950,65701094 (2002/08/08 18:15:01.094)

GENERIC_RECORD_HEADER for MDR at offset 40927 bytes:
   RECORD_CLASS 8 (MDR)
   INSTRUMENT_GROUP 0 (GENERIC)
   RECORD_SUBCLASS 0 (mdr-l0)
   RECORD_SUBCLASS_VERSION 1
   RECORD_SIZE 18764
   RECORD_START_TIME 950,65701469 (2002/08/08 18:15:01.469)
   RECORD_STOP_TIME 950,65701469 (2002/08/08 18:15:01.469)
```

```
GENERIC_RECORD_HEADER for MDR at offset 59691 bytes:
   RECORD_CLASS 8 (MDR)
   INSTRUMENT_GROUP 0 (GENERIC)
   RECORD_SUBCLASS 0 (mdr-l0)
   RECORD_SUBCLASS_VERSION 1
   RECORD_SIZE 18764
   RECORD_START_TIME 950,65701844 (2002/08/08 18:15:01.844)
   RECORD_STOP_TIME 950,65701844 (2002/08/08 18:15:01.844)

GENERIC_RECORD_HEADER for MDR at offset 78455 bytes:
   RECORD_CLASS 8 (MDR)
   INSTRUMENT_GROUP 0 (GENERIC)
   RECORD_SUBCLASS 0 (mdr-l0)
   RECORD_SUBCLASS_VERSION 1
   RECORD_SIZE 18764
   RECORD_START_TIME 950,65702219 (2002/08/08 18:15:02.219)
   RECORD_STOP_TIME 950,65702219 (2002/08/08 18:15:02.219)

GENERIC_RECORD_HEADER for MDR at offset 97219 bytes:
   RECORD_CLASS 8 (MDR)
   INSTRUMENT_GROUP 0 (GENERIC)
   RECORD_SUBCLASS 0 (mdr-l0)
   RECORD_SUBCLASS_VERSION 1
   RECORD_SIZE 18764
   RECORD_START_TIME 950,65702594 (2002/08/08 18:15:02.594)
   RECORD_STOP_TIME 950,65702594 (2002/08/08 18:15:02.594)

GENERIC_RECORD_HEADER for MDR at offset 115983 bytes:
   RECORD_CLASS 8 (MDR)
   INSTRUMENT_GROUP 0 (GENERIC)
   RECORD_SUBCLASS 0 (mdr-l0)
   RECORD_SUBCLASS_VERSION 1
   RECORD_SIZE 18764
   RECORD_START_TIME 950,65702969 (2002/08/08 18:15:02.969)
   RECORD_STOP_TIME 950,65702969 (2002/08/08 18:15:02.969)

GENERIC_RECORD_HEADER for MDR at offset 134747 bytes:
   RECORD_CLASS 8 (MDR)
   INSTRUMENT_GROUP 0 (GENERIC)
   RECORD_SUBCLASS 0 (mdr-l0)
   RECORD_SUBCLASS_VERSION 1
   RECORD_SIZE 18764
   RECORD_START_TIME 950,65703344 (2002/08/08 18:15:03.344)
   RECORD_STOP_TIME 950,65703344 (2002/08/08 18:15:03.344)

GENERIC_RECORD_HEADER for MDR at offset 153511 bytes:
   RECORD_CLASS 8 (MDR)
   INSTRUMENT_GROUP 0 (GENERIC)
   RECORD_SUBCLASS 0 (mdr-l0)
   RECORD_SUBCLASS_VERSION 1
   RECORD_SIZE 18764
   RECORD_START_TIME 950,65703719 (2002/08/08 18:15:03.719)
   RECORD_STOP_TIME 950,65703719 (2002/08/08 18:15:03.719)
```

… file trimmed at this point.