EUMETSAT

EUM FG 8

The Meteosat Archive

1978 1995 2000

FORMAT GUIDE No. 8

Cloud Analysis (CLA)

OpenMTP Format

Revision 1.1 - January 1998

Meteorological Archive and Retrieval Facility
EUMETSAT - Am Kavalleriesand 31 - D-64295 Darmstadt - Germany

Table of Contents

1. INTRODUCTION

1.1. Overview

The Cloud Analysis (CLA) product is a coded representation of the cloud in a single image, and is based directly on the cloud clusters extracted during segment processing.

This document describes the OpenMTP format for CLA product retrieval. This is a new format developed for the MTP programme, and represents an enhancement and evolution from the previous IBMMOP format used by the MOP programme up to mid-November 1995.

1.2. Contact Point

All enquiries relating to this document or to the Meteosat Archive in general should be directed to:

MARF Customer Enquiries
EUMETSAT
Am Kavalleriesand 31
64295 Darmstadt
Germany

Telephone: +49 6151 807 377

Telefax: +49 6151 807 379

Electronic mail: archive@eumetsat.de

2. OVERVIEW

2.1. Introduction

This section provides brief details of the background to the OpenMTP format for CLA products.

The OpenMTP format is a new format developed for the Meteosat Transition Programme (MTP). It represents a progression from the IBM-compatible 'IBMMOP' format used by ESOC during the preceding Meteosat Operational Programme (MOP) which ran until mid-November 1995.

The main differences between the OpenMTP and IBMMOP formats are as follows:

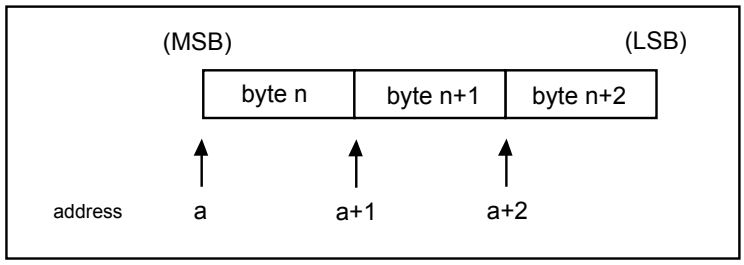
- The machine level representation of bits and bytes used in the OpenMTP format follows the standard used by UNIX / open systems architecture (SUN, HP, SGI ...) machines, whereas the IBMMOP format follows the standard used by IBM machines. The open systems representation uses the IEEE standard for real number representation, and ASCII rather than EBCDIC encoding for character data. It is anticipated that support for this open system representation will provide increased convenience for users.

  The OpenMTP data representation is discussed in section 2.2.
- The OpenMTP format includes an additional ASCII format header which can be easily examined by a user to check the content of a product file.
- The OpenMTP format provides significant extra information for products generated in the MTP era (i.e. since mid-November 1995). This information is stored in fields which are not present in the IBMMOP format.

2.2. Representation

This section describes the open system machine representation of the basic data types character, logical, short integer (two byte), integer (four byte), and single-precision floating-point.

The representation is 'big endian' which implies the following layout:



Where byte n is more significant than byte n+1. That is, the most significant byte is located at the lowest address, the least significant byte is located at the highest address. This is in contrast to little endian format (employed by for instance DEC VAXes and IBM PCs) where the least significant byte is located at the smallest address and the most significant bytes are located at the highest address.

In the following, bytes will be numbered from left to right starting with 0. Also bits are numbered from left to right starting with 0. Thus in a two byte integer, for example, the left-most byte will be given the number 0, the right-most byte will be given the number 1, the left-most bit will be given the number 0 and the right-most bit will be given the number 15.

Character type

Character fields are coded in ASCII and occupy 1 byte of storage.

Logical type

Logical fields are coded in single bytes. A byte value of 0 corresponds to 'FALSE' and any other value to 'TRUE', although in line with convention a value of 1 is normally used for 'TRUE' within the OpenMTP format.

Short integer

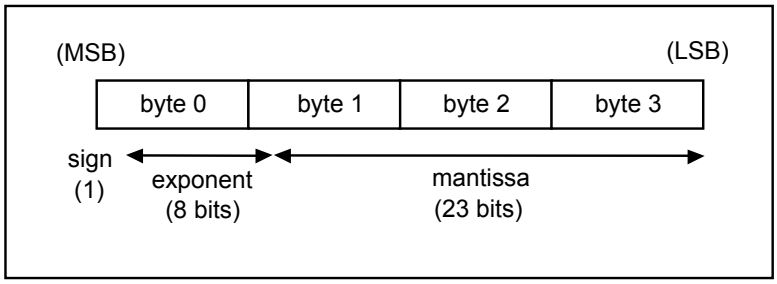
A short integer is two bytes in length. The short integer is represented in two's complement which means that bit 0 of byte 0 has negative weight (-bit0 * 2**15). Unless otherwise stated, short integer fields should therefore be interpreted as signed values with a range of -32768 ... 32767.

Integer type

A full integer is four bytes in length. It is represented in two's complement which means that bit 0 of byte 0 has negative weight (-bit0 * 2**31).

Single-precision floating point

A single-precision (four byte) floating point number has the following representation:



The following three fields describe the single-precision floating-point:

S: The sign of the number. Values 0 or 1 represent positive and negative respectively. One bit (bit 0) is devoted to this field.

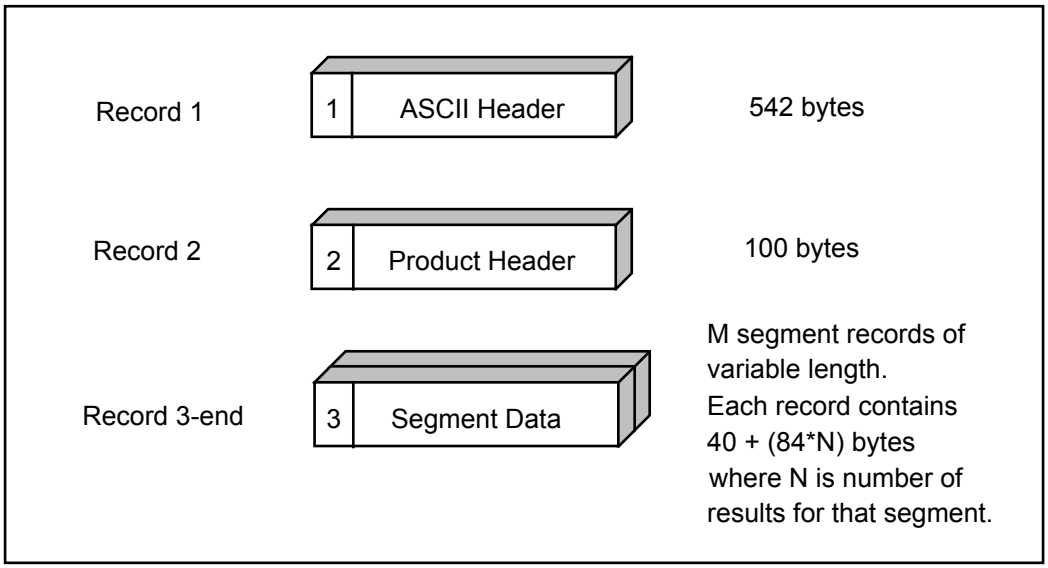
E: The exponent of the number, base 2. 8 bits are devoted to this field. The exponent is biased by 127. Thus the range of the exponent is -127 to 128.

M: The fractional part of the number's mantissa, base 2. 23 bits are devoted to this field. The integer part of the mantissa is always a binary 1 for which reason it is implicit in the representation.

3. PRODUCT STRUCTURE

The overall product structure is shown in figure 2. The product consists of a variable number of records of variable length.

Figure 2 CLA Product Structure



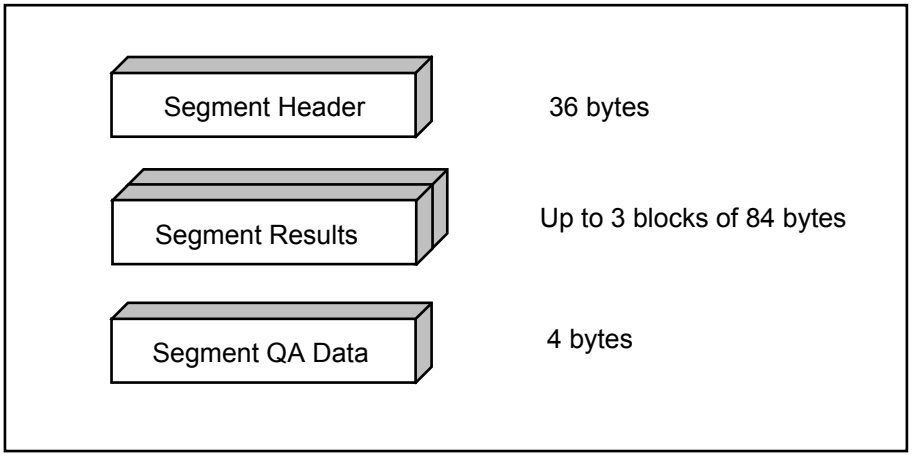
The structure contains three distinct components:

- Record 1, ASCII header. This fixed length (542 byte) record contains general information about the file in ASCII format. It enables a user to quickly check the content of the product using a basic editor or print function.
- Record 2, product header. This fixed length (100 byte) record contains binary format information relating to the overall content of the file.
- Records 3 onwards, CLA segment data. Contains the percentage coverage, temperature, and cloud top pressure for each of up to three identified cloud layers within the segment along with quality and other control information.

  The segment records are hierarchically structured as shown in figure 3. One record is included for each of the 'M' segments for which there is a CLA result (typically 2000 segments in all). Segments with no results (ie no detected cloud layers) are not included in the product. The number of segments present is recorded in record 2 to assist automatic parsing of the product structure.

  Each segment record in turn contains a 36 byte segment header, followed by up to three results for that segment, followed by 4 bytes of QA flags. Each result block corresponds to one identified cloud layer in the. Future algorithm enhancements may enable more than three layers to be recorded for each segment and the format has been designed to accommodate this. Again, the number of results is recorded in the segment header to assist automatic parsing of the product structure. Each result block occupies 84 bytes.

Figure 3 CLA Segment Data Structure



Assuming that one quarter of all segments are cloud free, one quarter have one cloud layer, one quarter have two cloud layers and one quarter have three cloud layers, the size of a typical CLA product is approximately 625 Kilobytes. The exact size may be defined as:

$$SIZE \quad = \quad 542 + 100 + \sum_{S=1}^{S=M} \left(36 + R_S \times 84 + 4\right) \qquad \text{bytes}$$

where S is the segment number, R_S the number of results for segment number S, and M the total number of segments with results in the product. This equation can be reduced to the expression:

$$SIZE \quad = \quad 642 + 40 \times M + 84 \times R \qquad \text{bytes}$$

where M is the number of segments with results and R is the number of results blocks (ie the total number of layers summed over all the segments) in the product.

The three types of record making up the CLA format have structures of type 1 to 3 respectively. Detailed descriptions of each structure are provided in section 4 of this document.

It should be noted that the 'records' in this product format are purely logical. The file should in practice be seen as a single structure consisting of a sequence of bytes.

4. FORMAT AND FIELD DEFINITIONS

This section provides detailed format definitions for each of the three structures introduced in the previous section.

The following information is provided for each field:

- Offset from start of structure. (To get the overall offset from the start of the file, this number must be added to N times 6444, where N is the number of preceding physical records). The offsets are quoted in zero-relative terms.
- Name of the field. An arbitrary but convenient field identifier.
- Description. Describes the field and any special features of its population.
- Type. The data type of the field, i.e. how it is encoded. The valid types are:

  A<n> - An ASCII string of <n> characters.

  B<n> - A string of <n> values to be treated as simple bytes.

  I2 - A 2-byte integer in binary format.

  I4 - A 4-byte integer in binary format.

  L1 - A one-byte logical value (TRUE or FALSE).

  R4 - A single-precision floating-point (4-byte real) number in binary format.

  See section 2.2 for detailed descriptions of the encoding of each type in the file.
- Dimension. The number of entries in the field, e.g. 1 for a single value, 10 for an array of 10 values, (10, 10) for a two-dimensional matrix of 10 rows of 10 values, etc. The first index quoted is that which cycles fastest, i.e. the first index cycles once for each step in the second index, etc.

Footnotes to each table provide additional information where necessary.

4.1. ASCII Header

As mentioned in section 3, header record 1 is a fixed length ASCII text block of 542 bytes. The record is divided into a series of text lines each of which has the same format, vis:

FIELD_NAME FIELD_VALUE <newline>

Every field starts with a field name, which describes the content of the field. The field name is padded out to 15 characters total width with spaces, and is left justified. The maximum length of the text is 14 characters, so that character 15 (dividing the field name from the field value) is always a blank.

The field value starts at character 16 of the field and continues until character N-1, where N is the total length of the field. If the value text does not extend to this character, the field will be padded with spaces.

A newline character is inserted at character position N of every field, so that a sensible line-by-line format is displayed when a user lists out the opening bytes of the product file using an editor or print command.

The fields of the ASCII header record are given in the table below. The indicated field lengths are the total lengths and therefore include the 15 characters used for the field name and the terminating newline character. The field names that appear in each field are noted as part of the description of the field.

Offset	Name	Description	Type	Dimension
000	PROD	Field name = 'Product'. MTP era name of product - set to 'CLA'.	A25	1
025	FORMAT	Field name = 'Format'. Name of data format, always set to 'OpenMTP'.	A55	1
080	FVERS	Field name = 'FormatVersion'. Version number of format for this file, initial value is '1'.	A75	1
155	PLTFRM	Field name = 'Platform'. Satellite name in free text format, e.g. 'Meteosat-5'.	A30	1
185	DATE	Field name = 'Date'. Nominal date of product in YYYY-MM-DD format, e.g. '1996-11-30'.	A26	1
211	TIME	Field name = 'NominalTime'. Nominal time of product in HH24:MI format, e.g. 10:59, 22:30.	A21	1
232	SLOT	Field name = 'SlotNo'. Slot number in day: 1 ... 48.	A19	1
251	ORDER	Field name = 'Ref'. Unique reference number of the file within the EUMETSAT order handling system, in ORDER-DELIVERY-ENTRY-ITEM format, e.g. 1767-1-2-10.	A47	1
298	CUST	Field name = 'Source'. Identifier of customer requesting product.	A35	1
333	PTIME	Field name = 'Time'. Production time in YYYY-MM-DD-HH24:MI format, e.g. 1996-11-30-14:30.	A35	1
368	SWVERS	Field name = 'SWVersion'. Software Version used for production.	A75	1
443	FNAME	Field name = 'FileName'. Identifier of data type in ESOC format, provided for compatibility and continuity. For CLA data this will always be set to ' CANI3AU'.	A24	1
467	CRIGHT	Field name = 'Copyright'. EUMETSAT Copyright notice.	A75	1

4.2. Product Header

This section defines the content of the 100 byte binary product header.

Offset	Name	Description	Type	Dimension
000	SLOT	Slot number (1-48).	I4	1
004	TIME	Slot nominal time in HH24MI format, e.g. 1030, 2200.	I4	1
008	JDAY	Day of the year (eg 123)	I4	1
012	YEAR	Year, e.g. 1996	I4	1
016	PLTFRM	Spacecraft identification, in Mx or METx format, i.e. 'M5' or 'MET5' for Meteosat-5.	A4	1
020	Spares	Spares (8 bytes).	--	--
028	FNAME	Product name, always set to ' CLA' for CLA products.	A4	1
032	PTIME	Product time.	I4	1
036	PALG	Product extraction algorithm.	A32	1
068	PVERS	Version of product (raw, final etc).	I4	1
072	NSEG	M, the number of segments with results in the product (typically approx. 2000)	I4	1
076	MQCFLG	MQC done flag	L1	1
077	Spares	Spares (15 bytes).	--	--
092	QTOTAL	Combined quality indicator for whole product	I4	1
096	DIST	Distribution authorization, logical flag set if product was authorised for distribution at time of generation.	L1	1
097	Spares	Spares (3 bytes).	--	--

4.3. CLA Data

The first part of the segment record consists of a 36 byte segment header. Every segment is identified by its row (line) and column number within the overall 80 x 80 segment grid.

Offset	Name	Description	Type	Dimension
000	SEGLIN	Segment line (1-80)	I4	1
004	SEGCOL	Segment column (1-80)	I4	1
008	SELPIX	South East corner line pixel number	I4	1
012	SECPIX	South East corner column pixel number	I4	1
016	SELAT	South East corner latitude in degrees.	R4	1
020	SELON	South East corner longitude in degrees.	R4	1
024	SHEIGHT	Segment height in pixels (currently 32).	I4	1
028	SWIDTH	Segment width in pixels (currently 32).	I4	1
032	NRES	N, the number of CLA results blocks in segment (currently can be set to 1, 2 or 3)	I4	1

The following part of the table describes the structure of the 84-byte results block. This block is repeated N times per segment. Offsets are given from the start of the segment structure for the first block; add 84 or 168 as appropriate to obtain the offsets for the equivalent parameters in the second and third blocks.

Offset	Name	Description	Type	Dimension
036	CENLAT	Latitude of segment centre	R4	1
040	CENLON	Longitude of segment centre	R4	1
044	CLA	Cloud layer amount (percentage cover of this layer within the segment).	R4	1
048	CLAT	Cloud layer temperature (degrees Celsius X 100).	R4	1

052	CLAP	Cloud layer top pressure.	R4	1
056	Spares	Spares (8 bytes).	--	--
The following fields contain parameter quality indicators.				
064	LOCQ	Location quality indicator (for possible future use)	I4	1
068	CLAQ	Cloud layer amount quality indicator.	I4	1
072	CLATQ	Cloud layer temperature quality indicator.	I4	1
076	CLAPQ	Cloud layer top pressure quality indicator.	I4	1
080	Spares	Spares (8 bytes).	--	--
The following fields will eventually contain quality indicators from the Automatic Quality Control (AQC) process.				
088	Spares	Spares for AQC quality indicators (32 bytes, not currently used).	--	--

The following four bytes contain quality control flags from the Automatic and Manual Quality Control processes. They appear at the end of the segment, i.e. one set of flags exists per segment. The offset position of these flags depends on NRES, the number of preceding results blocks.

Offset	Name	Description	Type	Dimension
36+84* NRES	AQCREJ	AQC rejected flag	L1	1
37+84* NRES	MQCREJ	MQC rejected/reinstated flag	L1	1
38+84* NRES	MQCMOD	MQC data modified flag	L1	1
39+84* NRES	Spares	Spare (1 byte).	--	--

5. ADDITIONAL NOTES

5.1. Applicability

The format description applies equally to products generated from the Meteosat Operational Programme (1978 – mid-November 1995) and from the Meteosat Transition Programme (mid-November 1995 onwards).

However, the format includes many fields added as an enhancement for MTP, which cannot be fully populated when MOP era data is retrieved as the required underlying data is not available from the archive.

5.2. Format History

The OpenMTP product format is an evolution from the IBMMOP format used for many years by ESOC within the MOP programme. Users who wish to continue with the ESOC format can still retrieve the data in that form. However, the OpenMTP format offers additional data and features which should enhance the value of the product to most users.

5.2.1. Evolution of Algorithms

The algorithms used to generate CLA data for archiving have inevitably evolved over the years. It is not possible to provide a detailed history, but key points to be considered when evaluating older data will be noted here in a future issue of this document.

5.2.2. Retrieving MOP Data in OpenMTP Format

There are many new fields in the OpenMTP format which cannot be populated for MOP era data from the Meteosat Archive. These fields, and the way they are populated when these older data are requested, are tabulated below. (Note that these limits only affect the binary header record and the segment records; the ASCII header is fully populated for all data).

Binary Header Record

Offset	Name	Value for MOP Era Data
016	PLTFRM	' N/A'. The original satellite from whose data the product was derived is not recorded in the ESOC archive.
036	PALG	'MIEC: Information Not Available'. The algorithm history is not available.
068	PVERS	Always set to 0 for MOP era data. (The version numbers for MTP era data start from 1).
076	MQCFLG	Set to FALSE (arbitrary).
092	QTOTAL	Set to 0 (arbitrary).
096	DIST	Set to FALSE (arbitrary).

CLA Segment Data Records

Offset	Name	Value for MOP Era Data
052	CLAP	Set to 0. Information not available for MOP era data.
064	LOCQ	Set to FALSE. Information not available for MOP era data.
068	CLAQ	Set to FALSE. Information not available for MOP era data.
072	CLATQ	Set to FALSE. Information not available for MOP era data.
076	CLAPQ	Set to FALSE. Information not available for MOP era data.
36+84* NRES	AQCREJ	Set to FALSE. Information not available for MOP era data.
37+84* NRES	MQCREJ	Set to FALSE. Information not available for MOP era data.
38+84* NRES	MQCMOD	Set to FALSE. Information not available for MOP era data.

5.3. Health Warnings

There are no known format errors.