Earth Explorer

Ground Segment

File Format Standard

Prepared by: Earth Observation Programs
System Support Division

Checked by: P. Viau

•
ESTEC
Noordwijk
The Netherlands
•

Change Record

Issue	Date	Page	Section	Req't	Description of Change
0.5	3 Oct 2001	All			initial draft Issue, limited distribution
0.6	30 Oct 2001	All			draft issue, distributed for comments
0.9	9 Nov 2001	All			draft issue, for Ground Segment Requ. Review
1.0	1 May 2002	All			updated after comments from all reviewers
1.1	24 May 2002	All			clarified file naming consolidated Fixed Header content added binary records description in XML Variable header
1.2	12 Jul 2002	All			consolidated binary records description in XML Variable Header (arrays, structures) added ASCII guideline on usage of time reference corrected variable XML structure representation clarified empty tags usage corrected syntax for lists tags
1.3	18 Oct 2002	All	All 2.2 3, 4 3, 4 6.3 7.1 4		cleaned up terminology: standard (not guidelines) added GOCE Master ICD as Reference Document clarified file naming vs packaging added ZIP as preferred packaging/compression tool added 4-bytes alignment for binary data corrected typos in Fixed Header example changed time format in file names (if used)
1.4	13 Jun 2003	All	All 5.2 7.2.3		small clarifications (see change bars) added validation schema reference for non-XML data removed record structure description from Variable Header

Table of Contents

1 Introduction

1.1 Background

The European Space Agency (ESA), referred to as "the Agency" in this document, is currently developing a set of Earth Explorer Missions within the Agency's Earth Observation Envelope Programme (EOEP).

These missions are small scale missions, with more compact schedules are smaller budgets than preceding Earth Observation Missions such as ERS and Envisat.

In order to promote a common approach for Ground Segment data exchanges within those missions, ESA is defining a file format standard, recorded in this document.

1.2 Purpose and Scope

This document contains the File Format Standard, relevant to all data files exchanged between ground segment systems within the Earth Explorer Missions.

The expected benefits of the common use of this standard are:

- increasing standardization of formats over several missions
- use of industry standards for data representation
- opportunity to use off-the-shelf, widely distributed tools for data handling and visualization
- reduced file design effort
- reduced file handling software development effort
- reduced test data production effort
- reduced range of integration problems, by virtually eliminating file syntax problems and focusing the effort on file semantics

The standard covers:

- file structure
- file naming
- file syntax
- headers content
- data representation
- availability of tools to support the standard

1.3 Standard Applicability

This standard must be followed for all files travelling between ground system systems, and should be considered also for files used within those systems.

The application of this standard will be strictly enforced for all systems developed under ESA responsibility, and will be encouraged as far as possible for other systems, not under ESA responsibility but participating in Earth Explorer Missions ground segments.

Note that a number of options are described in this document, for several aspects of the file formats. All documents formally describing file format details shall take into account this standard, and, for each file, clearly indicate the options selected. This is, for example, relevant for File Type naming, Variable Header structure, Data Block structure... (these terms are fully described in the following section).

Documents responsible for formal description of file format details are typically Interface Control Documents (ICDs). See [CS-ICD] for an overview of interfaces and documents which describe them, in the case of CryoSat.

1.4 Acronyms and Terminology

1.4.1 Acronyms

ADM	(also called ADM-Aeolus) Atmospheric Dynamics Mission satellite
ADP	Auxiliary Data Provider
Aeolus	see ADM
ALADIN	Atmospheric LAser Doppler INstrument (Aeolus instrument)
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CryoSat	Cryosphere Satellite
DORIS	Doppler Orbitography and Radio-Positioning Integrated by Satellite
EGG	Electrostatic Gravity Gradiometer (GOCE instrument)
EOEP	Earth Observation Envelop Programme
ESA	European Space Agency
ESL	Expert Support Laboratory
FH	Fixed Header
FOS	Flight Operations Segment
GOCE	Gravity and Ocean Circulation Explorer satellite
GPS	Global Positioning System
GSOV	Ground Segment Overall Validation
HK/TM	House-Keeping TeleMetry
HMI	Human-Machine Interface
ICD	Interface Control Document
ISP	Instrument Source Packet
MMI	Man-Machine Interface, see HMI
MPH	Main Product Header
LTA	Long-Term Archive
PDS	Payload Data Segment
POD	Precise Orbit Determination
SDE	Software Development Environment
SIRAL	Synthetic Interferometric Radar ALtimeter (CryoSat instrument)

SMOS	Soil Mositure and Ocean Salinity satellite
SPH	Specific Product Header
TBC	To Be Confirmed
TBD	To Be Defined
USF	User Services Facility
VH	Variable Header
Web	see WWW
WWW	World-Wide-Web
WYSIWYG	"What You See Is What You Get"(
XML	eXtensible Markup Language
XSL	XML Style Sheet

1.4.2 Terminology

This document and its appendixes use the following terms, briefly described here and fully defined in the relevant sections:

- Logical File — a set of data, consisting of a Header and a Data Block
- Physical File(s) — the computer file(s) storing a Logical File; one Logical File can be stored as a single Physical File, or as two separate Physical Files: the Header File and the Data Block File
- Header — the initial part of a Logical File, containing descriptive or configuration control information
- Data Block — the main part of a Logical File, containing the data
- Header File — a Physical File storing a Header
- Data Block File — a Physical File storing a Data Block
- File Name — the name of a file, which has a formal structure
- File Class — part of the File Name, defining the file usage context
- File Type — part of the File Name, uniquely defining the file structure
- File Category — sub-part of the File Type, used to define groups of files of similar nature
- File Instance ID — part of the File Name, defining specific information which might be needed for a given File Category; the size and content of the File Instance ID depends on the File Category
- Validity Period — consists of the pair (Validity Start Time, Validity Stop Time), defining the timing constraints of a file; can be part of the File Name, for some File Categories
- Validity Start Time — defines the start of the Validity Period
- Validity Stop Time — defines the end of the Validity Period
- Version Number — defining the file version; can be part of the File Name, for some File Categories

1.4.3 Document Conventions

This document uses specific fonts in thefollowing cases:

courier_bold_italic for file names, or elements of file name; these are not actual characters string, they indicate where actual strings would be located

courier_bold for strings actually used in file names, or elements of file name

courier_italic for file contents, or partial file contents; these are not actual file data, they indicate where actual data would be located

courier for actual file contents, or parts of file contents

2 Documents

2.1 Applicable Documents

No applicable documents have been identified.

2.2 Reference Documents

The following documents provide information referred to in this document.

[MCD]	Earth Explorer CFI - Mission Conventions Document	CS-MA-DMS-GS-0001
[GEN-SUM]	Earth Explorer CFI - General SUM	CS-MA-DMS-GS-0002
[IO-SUM]	Earth Explorer CFI - File IO Library SUM	CS-MA-DMS-GS-0007
[CS-ICD]	CryoSat Master ICD	CS-ID-ESA-GS-0147
[GO-ICD]	GOCE Master ICD	GO-ID-ESA-GS-0037
[XML]	XML Reference	(TBD)
[IEEE]	IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std 754-1985 (IEEE 754), Institute of Electrical and Electronics Engineers, 1985	

3 File Structure

3.1 Logical vs Physical Files

Each "Logical File" represents a set of data, which are logically organized hierarchically as follows:

- a Header
- a Data Block

In terms of computer "Physical Files", such a "Logical File" can be structured as either:

- a single Physical File
- two separate Physical Files:
  - a Header File
  - a Data Block File

The advantadge of separating Header and Data Block is in principle to facilitate processing of the files, which have very different contents and are processed by different software tools:

- Header: configuration control or organizational data
- Data Block: scientific data

NOTES:

- See Section 9.5 for tools allowing Header and Data Block splitting/joining

3.2 Header

The Header shall be structured in several parts:

- a Fixed Header (FH), with identical structure for all files
- a Variable Header (VH), which allows to define and structure different information for each file type

Note that file types which share some common header information, such as scientific data products, should structure the VH accordingly. For example, the VH of scientific data products shall contain:

- a Main Product Header (MPH)
- a Specific Product Header (SPH)

3.3 Data Block

The Data Block shall contain 1 or more Data Sets. Each Data Set should contain a number of Data Records, preferably of identical structure (i.e. same set of parameters, no variable structures).

Data Blocks should be kept as simple as possible:

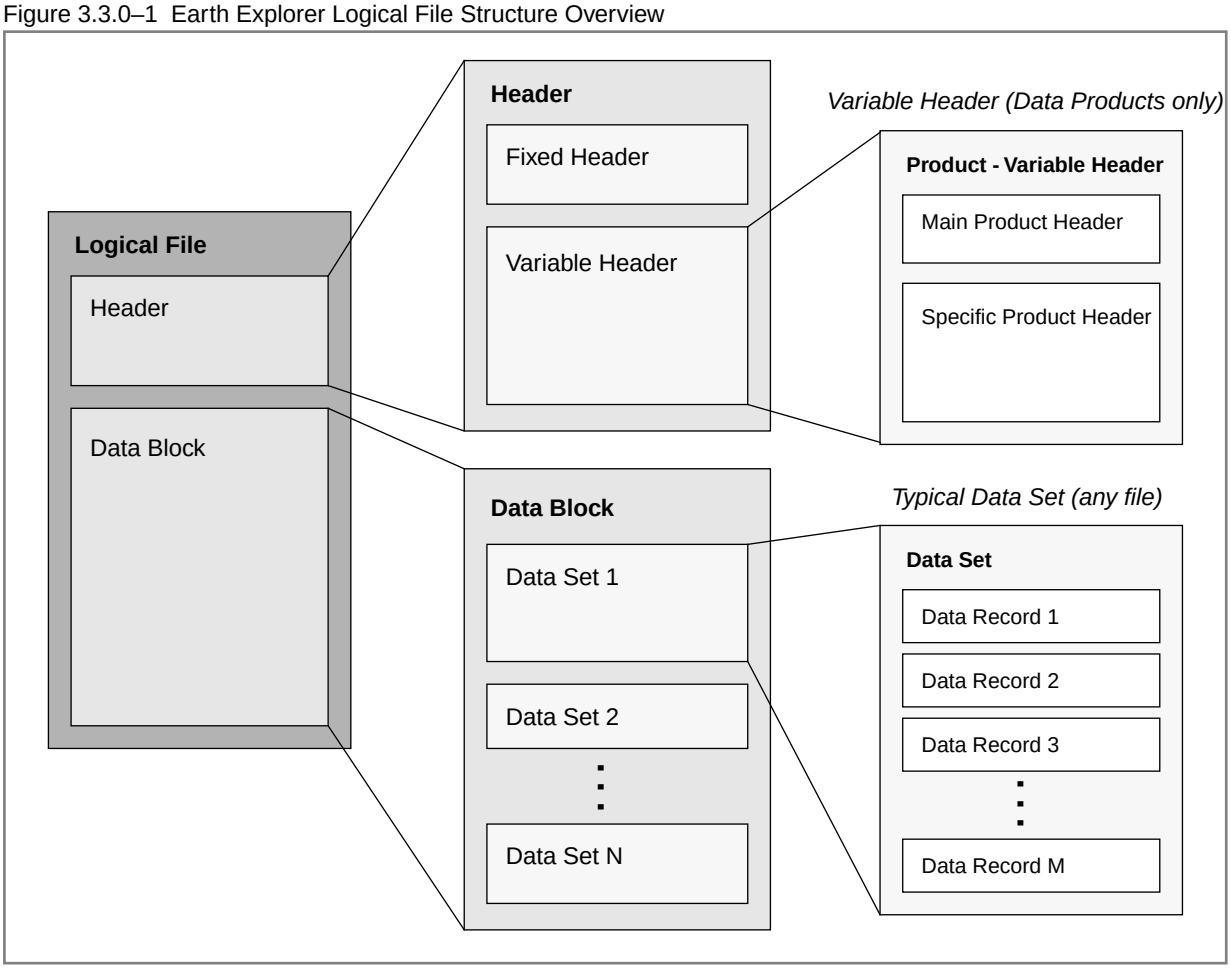
1. if possible, only 1 Data Set shall be used, with Data Records of identical structure
2. if not possible, it is acceptable to use several Data Sets; different Data Sets can have different Data Record structure, but within a Data Set all Data Records shall have identical structure

3. Data Set with variable Data Record structure should be avoided as far as possible

NOTE:

This describes the general case for Data Blocks. There can be exceptions to the Data Block structures (see Section 5.2.3 and Section 8).

Figure 3.3.0–1 Earth Explorer Logical File Structure Overview



3.4 Packaging and Distribution of Files

When files are distributed, several considerations have to be taken into account, whatever the distribution media (CD-rom, DVD-rom, electronic links,...):

- file compression may be used for data transfer or media space usage efficiency
- if Header and Data Block are stored as separate Physical Files, it is vital that both files are packaged and distributed together

The allowed mechanisms used are:

- compression using the “gzip” format
- packaging several files using the “tar” format
- packaging and compression using the “zip” format

NOTES:

- the gzip and tar formats are originated from Unix, but are supported on most operating systems
- the zip format is originated from Windows, but is supported on most operating systems
- the packaging has the following advantadge: it could also be used to distributed, together with the data file(s), supporting files such as schemas (to validate the file format) and style-sheets (to visualize the file in a friendly, customized way); see Section 9 for more details on this

4 File Naming

4.1 Logical File Name

The files shall be named using a fixed set of elements, each of fixed size, separated by underscores "_".

Ex. 4.1.0–1 Earth Explorer Logical File Name

MM_CCCC_TTTTTTTTTT_<*instance ID*>

These elements constitute the smaller set of information which ensure that each Logical File name is unique, within the context of Earth Explorer ground segments.

The file name elements are described in the following table.

Table 4.1.0–1 Earth Explorer Logical File Name Elements

Naming Element	Description	Comment
MM	Mission ID, for example: - **CS** for CryoSat - **GO** for GOCE	2 uppercase letters
CCCC	File Class, i.e. the type of activity for which the file is used. Examples include: - **SVTx** for SVT tests (x = 0, 1, 2, 3) - **TDxx** for processing Test Data Sets (xx = 00..99) - **OPER** for routine operations - **TEST** for internal tests	4 uppercase letters, can contain digits
TTTTTTTTTT	File Type. Examples include: - **MPL_ORBREF**: reference orbit definition - **DOR_NAV_0_**: DORIS Navigator Level-0 - **SIR_ELE_2_**: SIRAL elevations Level-2	10 uppercase letters, can contain digits and underscores "_".
<instance ID>	Instance ID, making the file name unique. This part is variable in size and contents, and depends of the File Category (see below). Possible examples include: ***yyyymmdd*T*hhmmss_YYYYMMDD*T*HHMMSS_vvvv*** (Validity Period and Version Number) ***YYYYMMDD*T*HHMMSS_oooooooo*** (Creation Date and Orbit Number)	Maximum 41 characters, can contain uppercase letters, digits and underscores "_".

NOTES:

- all letters are uppercase, to avoid problems with some operating systems
- this section describes the Logical File Name convention; Physical File names are described in Section 4.2

4.1.1 Mission ID

Consists of 2 characters, all uppercase letters.

The list of authorized Mission IDs is fixed by this document:

Table 4.1.1–1 File Name - Mission IDs

Mission ID	Mission Name (for Fixed Header)
CS	CryoSat
GO	GOCE
AE	Aeolus
SM	SMOS
	others to be added when relevant

4.1.2 File Class

Consists of 4 characters, either uppercase letters or digits.

This element allows to specify the usage of the file, for a specific phase of the ground segment development or operations cycle.

It allows, in particular, to reset version counters for each new phase without any risk of having ambiguous file names. For example, mission planning files used for SVT tests can be numbered independently for each SVT test, and all of those can be independant from the routine operations numbering.

A typical list of File Classes could be:

- **SVTx** (x being 0, 1, 2, 3...) for SVT tests
- **TDxx** (xx being 00...99) for Test Data Sets
- **GSOV** for Ground Segment Overall Validation tests
- **OPER** for routine operations
- **TEST** for internal tests
- others TBD

Each Mission shall define the list of authorized File Classes.

For CryoSat, the official list of File Classes can be found in [CS-ICD].

For GOCE, the official list of File Classes can be found in [GO-ICD].

4.1.3 File Type

Consists of 10 characters, either uppercase letters or digits or underscores "_".

This element uniquely defines the file structure. All files of the same File Type share the same structure.

Each Mission shall define the list of authorized File Types.

For CryoSat, the official list of File Types can be found in [CS-ICD].

For GOCE, the official list of File Types can be found in [GO-ICD].

The following sub-sections describe the File Category, a vital part of the File Type, and the standard for instrument data products File Types.

4.1.3.1 File Category

Within the File Type, to organize the files properly, the initial 4 characters shall be reserved for a File Category sub-element.

The File Category sub-element should always include 3 characters and an ending under-score, to improve readability. Typical File Categories could be:

- **MPL**_ for mission planning files
- **TLM**_ for telemetry retrieval files
- **AUX**_ for auxiliary data files
- **III**_ for data products files, where III is instrument-specific (see Section 4.1.3.2)
- others TBD

Each Mission shall define the list of authorized File Categories.

For CryoSat, the official list of File Categories can be found in [CS-ICD].

For GOCE, the official list of File Categories can be found in [GO-ICD].

IMPORTANT NOTES:

- The File Instance ID (see below) is variable in size and content. Each File Category can be assigned a separate File Instance ID “shape” (i.e. all files of the same File Category have the same File Name “shape”).
- Of course the number of different File Instance ID “shapes” shall be limited to the minimum, to avoid unnecessary complexity. Several File Categories can share the same File Instance ID “shape”.

4.1.3.2 Data Products File Category and File Type

Data products File Types shall follow the standard as any File Type, but shall be further modelled as follows.

For data products, the File Type shall be structured as **`III_XXXXLL`**, where:

- **`III_`** is the File Category, which is instrument-specific, e.g.:
  - **`DOR_`** for DORIS
  - **`GPS_`** for GPS
  - **`STR_`** for Star Tracker, etc...)
  - **`SIR_`** for SIRAL (CryoSat instrument)
  - **`EGG_`** for EGG (GOCE instrument)
  - **`ALD_`** for ALADIN (Aeolus instrument)
  - etc...
- **`XXXX`** shall provide semantic description, e.g.:
  - **`NAV_`** for Navigator (a DORIS product)
  - **`LRM_`** for Low Rate Mode (a SIRAL mode)
  - **`ATEL`** for Along-Track Elevation (a SIRAL product)
- **`LL`** for the product level, e.g.:
  - **`0_`** for Level-0
  - **`1b`** for Level-1b
  - **`2_`** for Level-2

With the above definitions, examples of data product File Types could be:

- **`DOR_NAV_0_`** for DORIS Navigator Level-0
- **`SIR_LRM_1b`** for SIRAL Low Rate Mode Level-1b
- **`SIR_ATEL2_`** for SIRAL Along-Track Elevations Level-2

4.1.4 File Instance ID

This element must ensure that each File Name is unique. In addition, it shall provide useful information on that file.

Because ground segment files are of very diverse nature, and come from different sources, it is necessary to provide some flexibility to achieve unique and meaningful naming structures.

For this reason, File Instance IDs can have different “shapes”, i.e. different sizes and content.

In order to facilitate the identification of the File Instance ID “shape”, and to keep control of the number of different “shapes”, all files of the same File Category shall have the same File Instance ID “shape”. Of course, several File Categories can share the same File Instance ID shape.

Each Mission shall define the File Instance ID “shape” for each File Category.

For CryoSat, the official list of File Instance ID “shapes” can be found in [CS-ICD].

For GOCE, the official list of File Instance ID “shapes” can be found in [GO-ICD].

NOTES:

File Instance ID “shapes” shall have a maximum of 41 characters, to respect File Name size constraints (see Section 4.1.6).

A typical Mission should have:

- 5 to 10 File Categories
- 1 to 3 File Instance ID shapes

Any File Instance ID "shape" should include at least one "date" element, typically Creation Date or Validity Start and/or Stop Time.

Examples of typical File Instance ID "shapes" are (see sub-elements definition below):

- ***yyyymmdd*T*hhmmss*_*YYYYMMDD*T*HHMMSS*_*vvvv*** (i.e. Validity Period and Version)
- ***YYYYMMDD*T*HHMMSS*_*oooooooo*** (i.e. Creation Date and Orbit Number)

4.1.5 Possible File Instance ID Sub-Elements

The following sub-sections describe typical File Instance ID sub-elements.

4.1.5.1 Creation Date

The Creation Date consists of:

- 8 characters, all digits, for the date: "***yyyymmdd***"
- 1 uppercase T: "**T**"
- 6 characters, all digits, for the time: "***hhmmss***"

NOTE:

- this format is defined in [MCD] as "CCSDS compact"
- the time format is designed such that times can be easily sorted chronologically

4.1.5.2 Validity Start and Stop Times (Validity Period)

The Validity Period is defined by a (Validity Start Time, Validity Stop Time) pair.

Each of these validity times consists of:

- 8 characters, all digits, for the date: "***yyyymmdd***"
- 1 uppercase T: "**T**"
- 6 characters, all digits, for the time: "***hhmmss***"

Depending on the File Type, the Validity Period can have a slightly different usage. For example:

- for reference mission planning files, the Validity Period indicates over which time period the file is valid
- for command sequences files, the Validity Period indicates at what time the commands must be executed
- for data products, the Validity Period indicates when the data were acquired
- for telemetry retrieval files, the Validity Period indicates when the telemetry data were requested to be retrieved
- others TBD

Each Mission shall define, for each File Type, the exact usage of the Validity Period.

NOTES:

- the time format is designed such that times can be easily sorted chronologically
- this format is defined in [MCD] as "CCSDS compact"
- 2 special validity times are defined (as specified in [MCD]):
  - **"00000000T000000"** for beginning of mission
  - **"99999999T999999"** for end of mission
- files for which the validity period is irrelevant shall use the beginning and end of mission validity times
- all times shall be in UTC time reference, as defined in [MCD]

4.1.5.3 *Version Number*

Consists of N characters, all digits.

A new version number should only be needed when all preceeding File Name elements, including the Validity Period, are unchanged.

Note that for some File Types, this strategy may not be sufficient to distinguish all files (e.g. for sets of files of the same File Type, generated together and used together). In that case the use of other File Instance ID sub-elements is recommended to remove the ambiguity.

NOTES:

- number shall start at 1 (not 0)
- 4 digits Version Number would be enough for 30 years of operations for file types resulting in 1 new version per day.
- 6 digits Version Number would be enough for 20 years of operations for file types resulting in 10 new versions per orbit.
- anyway such file types, requiring new files every day / orbit, should normally use different Validity Periods, thus not requiring to also change the Version Number; usage of Validity Period in the File Name removes the need to have a large Version Number field

4.1.6 File Name Sizes

The size of File Names depend on the size of the File Instance ID, but all File Names must be smaller than 64 characters.

This minimizes the problems of file name truncation on some operating systems, and in particular on CD-rom file systems.

4.2 Physical File Names

4.2.1 File Names and Extensions

All Physical Files related to the same Logical File shall share the file name, only differentiating each Physical File using a different extension.

Ex. 4.2.1–1 Earth Explorer Physical File Name

MM_CCCC_TTTTTTTTTT_<*instance ID*>.*XXX*

where "***XXX***" is a different extension depending on the physical file. See Section 4.2.2 and Section 4.2.3 for possible extensions.

4.2.2 Single File vs Header and Data Block Files

The following Physical File Names shall be used:

- **MM_cccc_TTTTTTTTTT_<*instance ID*>.EEF**
  for a complete file (containg both Header and Data Block)
- **MM_cccc_TTTTTTTTTT_<*instance ID*>.HDR**
  for Header file
- **MM_cccc_TTTTTTTTTT_<*instance ID*>.DBL**
  for a Data Block file

4.2.3 Packaging and Distribution of Files

To support packaged files, the following utilities can be used:

- "zip"
- "tar" and/or "gzip"

Note that "zip" is the preferred mechanism, because, as opposed to "tar/gzip":

- it compresses individual files before packaging them
- it therefore allows quicker retrieval of individual files within a compressed package
- software APIs exist to manipulate zip package from within an application

When using packaged files, to preserve file naming consistency and traceability, the following names shall be used.

Using "zip":

- **MM_cccc_TTTTTTTTTT_<*instance ID*>.ZIP**
  for a "zip" package containing either:
  – a complete file ("**.EEF**") after compression using "zip"
  – several files (typically "**.HDR**" and "**.DBL**") after compression using "zip"
  Note that in both cases, the package can also contain supporting files (see notes below)

Using "tar" and/or "gzip":

- **MM_cccc_TTTTTTTTTT_<*instance ID*>.GZ**
  for a complete file ("**.EEF**") after compression using "gzip"
- **MM_cccc_TTTTTTTTTT_<*instance ID*>.TAR**
  for a "tar" package containing several files (typically "**.HDR**" and "**.DBL**", but see also notes below)
- **MM_cccc_TTTTTTTTTT_<*instance ID*>.TGZ**
  for a "tar" package ("**.TAR**") after compression using "gzip"

NOTES:

- the gzip and tar formats are originated from Unix, but are supported on most operating systems
- the zip format is originated from Windows, but is supported on most operating systems
- the packaging has the following advantadge: it could also be used to distribute, together with the data file, supporting files such as schemas (to validate the file format) and style-sheets (to visualize the file in a friendly, customized way); see Section 9 for more details on this

5 File Syntax

5.1 General Considerations

5.1.1 Standard File Syntax - XML

All files should be coded in ASCII, following the XML-based syntax specified in this document.

Version 1.0 of the XML specification shall be used.

5.1.2 Exceptions

Acceptable reasons for not following the ASCII XML syntax, to be investigated and agreed with ESA on a case-by-case basis, are:

- files are too large (e.g. 10 Mbytes or more) and the overhead of using ASCII rather than binary is not acceptable
- files (ASCII or binary) are already designed and file production software is already implemented; note, however, that file conversion to the standard XML-based syntax should be investigated

For files which are accepted as exceptions, the following strategy shall apply:

- the original file, untouched, shall be placed in the Data Block (see Section 5.2.3)
- a proper Header shall be generated

The decision of which system, between the sender and the recipient of the file, creates the Header and/or merges Header and DataBlock, shall be negotiated on case-by-case basis and formally recorded in the ICD between the 2 systems.

Note that, referring to file naming standard defined in Section 4.1 and Section 4.2:

- if the sender only handles the original file (non XML-based), the file shall use the Data Block name style, ***file_name*.DBL**
- if the sender also creates the Header, as a separate file, the Header file shall use the Header name style, ***file_name*.HDR**, and Header and Data Block files shall be sent together in a package (preferably ***file_name*.ZIP**, but see Section 4.2.3 for alternatives)
- if the sender creates the Header and merges Header and Data Block, the file shall use the complete file name style, ***file_name*.EEF**
- optional compression can be used before sending the single file (preferably ***file_name*.ZIP**, but see Section 4.2.3 for alternatives)

5.1.3 Earth Explorer XML Conventions

XML has a very wide range of applications, and in that respect is a very flexible descriptive language. This section provides conventions applicable to the usage of XML for Earth Explorer files, which allow to restrict that flexibility and therefore simplify the usage of XML (see [XML] for a complete XML reference).

5.1.3.1 Basic Tags Conventions

The basic XML principle is that data are represented values enclosed within tags. Tags can have one or more optional attributes. The syntax is shown below.

Ex. 5.1.3–1 Basic XML Tag Syntax

```
<tag_name attribute_name="attribute_value">value</tag_name>
```

A few practical examples are given below.

Ex. 5.1.3–2 Examples of XML Tags

```
<flower>rose</flower>
<manager company="BBC" department="News">John DOE</manager>
<Longitude unit="deg">+180.000</Longitude>
```

Note that XML also allows "empty tags", where data are provided using attributes only, using the syntax `<tag_name attribute_name="attribute_value"/>`. This form is not relevant for Earth Explorer files.

Caution: the "empty tag" is different from a basic tag with empty value, using the syntax `<tag_name></tag_name>` or `<tag_name/>`.

The following specific conventions for basic tags shall apply to Earth Explorer XML ASCII files:

xml-1 XML "empty tags" shall not be used (i.e. the "empty tag" form, as described above)

xml-2 XML tags with empty value are allowed, but should be avoided as far as possible; the preferred representation is `<tag_name></tag_name>`

xml-3 XML tags and attributes shall use only letters, digits and underscores "_"

xml-4 XML tags and attributes should be full names or acronyms, unambiguously readable

xml-5 XML tags and attributes can consist of one or more words, separeted by underscores "_"

xml-6 Each XML tag word shall start with an uppercase letter (e.g. `Word1_Word2`), except for "linking words" such as "of" (e.g. `List_of_Things`)

xml-7 Attributes letters shall all be lowercase

xml-8 The use of attributes shall be limited to a few cases; usable attributes are:

- `unit` , always to be used for numerical values, if relevant (some values have no units)
- `count` , always to be used for lists of identical elements, TBC (see Section 5.1.3.2)
- `structure` , always to be used for tags of variable structures (see Section 5.1.3.2)
- `type` , always to be used for Data Blocks (see Section 5.2.3)
- `byte-order` , always to be used for Data Blocks (see Section 5.2.3.3)

5.1.3.2 *Hierarchical Structures Conventions*

The other basic XML principle is that data can be organized hierarchically, by embedding lower-level tags in higher-level tags. It is allowed to repeat tags, therefore creating lists. A repeated higher-level tag always contains the same sequence of lower-level tags, but some tags can be declared optional. The syntax is shown below.

Ex. 5.1.3–3 Basic XML Hierarchical Data Organization

```
<High_Level_Group_Tag>
      <Medium_Level_Tag_1>value_1</Medium_Level_Tag_1>
      <Medium_Level_Tag_2>value_2</Medium_Level_Tag_2>
      <Medium_Level_Group_Tag>
            <Low_Level_Tag_3>value_3</Low_Level_Tag_3>
            <Opt_Low_Level_Tag_4>value_4</Opt_Low_Level_Tag_4>
      </Medium_Level_Group_Tag>
      <Medium_Level_Group_Tag>
            <Low_Level_Tag_3>value_3</Low_Level_Tag_3>
      </Medium_Level_Group_Tag>
</High_Level_Group_Tag>
```

The following specific conventions for embedded tags shall apply to Earth Explorer XML ASCII files:

xml-9 Lists of identical data structures shall be embedded within a “list tag”

xml-10 List tags names shall use the syntax (attribute is TBC):

`<List_of_`*`element_name`*`s count="`*`number_of_elements`*`">`

xml-11 The higher-level list tag shall only contain a list of identical lower-level tags, each using the syntax <*`element_name`*> (for a list named `<List_of_`*`element_name`*`s>`)

NOTE: pleate notice the use of “s” after *`element_name`* in `List_of_`*`element_name`*`s.`

xml-12 Optional elements should not be used; acceptable cases for optional elements are:

- descriptive comments not affecting the data structures

xml-13 The structure (i.e. sequence of lower-level tags) within a given higher-level tag shall always be the same

xml-14 Exception: to allow some structural flexibility, it shall be possible to use variable structure, by using several choices of tag at a given location in the structure; "choice tags" shall be identified using the syntax:

```
<tag_name choice_of="variable_structure_id">
```

xml-15 All "choice tags" of a given variable structure shall use different tag names, but always use the same *"variable_structure_id"*

xml-16 A given tag shall be always be of either fixed structure or variable structure (i.e. it cannot have both forms)

xml-17 The structure of each choice of a variable structure shall be always be the same

xml-18 Variable structure should be avoided as far as possible

xml-19 Although lements at different locations in a hierarchical structure (e.g. `Medium_Level_Tag_1` and `Low_Level_Tag_3` in Ex. 5.1.3–3) can formally have the same tag name according to the XML specification, this should be avoided since it can create confusion.

Ex. 5.1.3–4 Earth Explorer File - XML List Syntax

```
<List_of_element_names count="number_of_elements">
      <element_name>element_content_1</element_name>
      <element_name>element_content_n</element_name>
      ...
      <element_name>element_content_N</element_name>
</List_of_element_names>
```

5.2 File Syntax - Hierarchical Decomposition

5.2.1 Top-Level File Syntax

For a given file of name ***file_name*.EEF**:

Ex. 5.2.1–1 Earth Explorer File - Complete File Syntax

```
<?xml version="1.0" ?>
<Earth_Explorer_File>
      Header  (same as the Header file file_name.HDR)
      Data Block
</Earth_Explorer_File>
```

See the following sections for Header and Data Block syntax.

NOTE:

- it is possible to keep the non-XML ASCII Data Block as a separate file, as long as header and data block are always transferred together as a package (see Section 4.2.3)
- the XML Prolog is used to control XML specification version; it shall be used for every XML file, including "**.EEF**", "**.HDR**" and "**.DBL**" if relevant

5.2.2 Header Syntax

This is identical to the Header file of name ***file_name*.HDR** (except that the Header file shall add the XML Prolog as initial line):

Ex. 5.2.2–2 Earth Explorer File - Header Syntax

```
<Earth_Explorer_Header Validation-Schema-Reference>
      <Fixed_Header>
            Fixed Header contents
      </Fixed_Header>
      <Variable_Header>
            Variable Header contents
      </Variable_Header>
</Earth_Explorer_Header>
```

NOTE:

- a reference to the XML schema reference allowing to validate the Header syntax shall always be provided as attribute to the Header tag; see Section 9.1 for XML validation tool details

5.2.3 Data Block Syntax

5.2.3.1 XML ASCII Data Block Syntax

Ex. 5.2.3–3 Earth Explorer File - XML ASCII Data Block Syntax

```
<Data_Block type="xml" Validation-Schema-Reference>
      Data Set 1
      Data Set 2
      ...
      Data Set N
</Data_Block>
```

The complete data block, including the <Data_Block> terminators, is identical to the Data Block file ***file_name*.DBL** (except that the Data Block shall add the XML Prolog as initial line).

NOTE:

- a reference to the XML schema reference allowing to validate the Data Block syntax shall always be provided as attribute to the Data Block tag; see Section 9.1 for XML validation tool details

5.2.3.2 Non-XML ASCII Data Block Syntax

Ex. 5.2.3–4 Earth Explorer File - Non-XML ASCII Data Block Syntax

```
<Data_Block type="ascii">
      <![CDATA[     (syntax TBC)
      Data Block Contents  (same as the Data Block file file_name.DBL)
      ]]>                  (syntax TBC)
</Data_Block>
```

This allows standard XML tools to properly parse and display the Data Block, although the Data Block contents will be shown exactly as written in the original, non-XML file.

Only the Data Block contents, without the <Data_Block> and <![CDATA]> terminators, is identical to the Data Block file ***file_name*.DBL**.

5.2.3.3 *Binary Data Block Syntax*

In the case of binary Data Blocks, the actual Data Block contents shall always be stored as a separate Data Block file (***file_name*.DBL**). This is an exception to the general file syntax as shown in Ex. 5.2.1–1, see example in Ex. 5.3.3–5.

Note, however, that standard XML tools will not be able to parse and display the Data Block, and ad-hoc tools must be used.

There is no way at this point in time to actually display binary data with XML tools.This is why Header and Data Block are kept separate, to avoid that failure to read the Data Block prevents from reading the Header with XML tools.

5.2.4 XML ASCII Data Set Syntax

Ex. 5.2.4–5 Earth Explorer File - XML ASCII Data Set Syntax

```
<data_set_name>
      Data Set Contents
</data_set_name>
```

The Data Set tag name can be any name respecting the tag name conventions (see Section 5.1.3.1).

Note that, usually, Data Sets should be lists of identical elements (see syntax in Ex. 5.1.3–4), for example to describe time-variant records of data.

Another form of Data Set consisting of a sequence of elements can also be used, for example for configuration files.

Whichever form is used, XML ASCII Data Sets shall use the standard XML syntax, while respecting the Earth Explorer XML Conventions, as specified in Section 5.1.3.

5.3 File Syntax - Summary

5.3.1 XML ASCII File Syntax

Ex. 5.3.1–1 Earth Explorer File - XML ASCII File Syntax - Complete File ("**.EEF**")

```
<?xml version="1.0" ?>
<Earth_Explorer_File>
      <Earth_Explorer_Header Validation-Schema-Reference>
            <Fixed_Header>
                  Fixed Header contents
            </Fixed_Header>
            <Variable_Header>
                  Variable Header contents
            </Variable_Header>
      </Earth_Explorer_Header>
      <Data_Block type="xml" Validation-Schema-Reference>
            <data_set_name_1>
                  Data Set 1 Contents
            </data_set_name_1>
            <data_set_name_2>
                  Data Set 2 Contents
            </data_set_name_2>
            ...
            <data_set_name_N>
                  Data Set N Contents
            </data_set_name_N>
      </Data_Block>
</Earth_Explorer_File>
```

Ex. 5.3.1–2 Earth Explorer File - XML ASCII File Syntax - Header & Data BlockFiles ("**.HDR**" & "**.DBL**")

Header File (***file_name*.HDR**)*:*

```
<?xml version="1.0" ?>
<Earth_Explorer_Header Validation-Schema-Reference>
      <Fixed_Header>
            Fixed Header contents
      </Fixed_Header>
      <Variable_Header>
            Variable Header contents
      </Variable_Header>
</Earth_Explorer_Header>
```

Data Block File (***file_name*.DBL**)*:*

```
<?xml version="1.0" ?>
<Data_Block type="xml" Validation-Schema-Reference>
      <data_set_name_1>
            Data Set 1 Contents
      </data_set_name_1>
      <data_set_name_2>
            Data Set 2 Contents
      </data_set_name_2>
      ...
      <data_set_name_N>
            Data Set N Contents
      </data_set_name_N>
</Data_Block>
```

Note that if Header and Data Block contents are stored as separate files, they shall be distributed as a package (preferably ***file_name*.ZIP**, but see Section 4.2.3 for alternatives).

5.3.2 Non-XML ASCII File Syntax

Ex. 5.3.2–3 Earth Explorer File - Non-XML ASCII File Syntax - Complete File ("**.EEF**")

```
<?xml version="1.0" ?>
<Earth_Explorer_File>
      <Earth_Explorer_Header>
            <Fixed_Header>
                  Fixed Header contents
            </Fixed_Header>
            <Variable_Header>
                  Variable Header contents
            </Variable_Header>
      </Earth_Explorer_Header>
      <Data_Block type="ascii">
            <![CDATA[(syntax TBC)
            Data Block Contents (same as file file_name.DBL)
            ]]>      (syntax TBC)
      </Data_Block>
</Earth_Explorer_File>
```

Ex. 5.3.2–4 Earth Explorer File - Non-XML ASCII File Syntax - Header & Data BlockFiles ("**.HDR**" & "**.DBL**")

```
Header File (file_name.HDR):
<?xml version="1.0" ?>
<Earth_Explorer_Header Validation-Schema-Reference>
      <Fixed_Header>
            Fixed Header contents
      </Fixed_Header>
      <Variable_Header>
            Variable Header contents
      </Variable_Header>
</Earth_Explorer_Header>

Data Block File (file_name.DBL): ad-hoc ASCII syntax
```

Note that if Header and Data Block contents are stored as separate files, they shall be distributed as a package (preferably ***file_name*.ZIP**, but see Section 4.2.3 for alternatives).

5.3.3 Binary File Syntax

Ex. 5.3.3–5 Earth Explorer File - Binary File Syntax

Header File (***file_name*.HDR**)*:*

```
<?xml version="1.0" ?>
<Earth_Explorer_Header Validation-Schema-Reference>
      <Fixed_Header>
                  Fixed Header contents
      </Fixed_Header>
      <Variable_Header>
                  Variable Header contents
      </Variable_Header>
</Earth_Explorer_Header>
```

Data Block File (***file_name*.DBL**)*: ad-hoc binary syntax*

Note that Header and Data Block contents are always stored as separate files, and shall be distributed as a package (preferably ***file_name*.ZIP**, but see Section 4.2.3 for alternatives).

6 Data Representation

6.1 General Considerations

The following data representation standard is aimed at improving standardization of units and data formats used, and, in the case of ASCII data, readability.

6.2 ASCII Data Representation

The following rules shall apply to ASCII data:

ascii-1 ASCII end of lines shall be marked only by the "new line" character (ASCII code 10), and NOT (accompanied) by the "carriage return" character (ASCII code 13). This is valid both for XML and non-XML ASCII data representation.

NOTE: the systematic use of ASCII end of lines after 1 or more XML tags is advised, although not strictly mandatory in the XML standard, in order to avoid long lines which can cause problems on some computer platforms.

ascii-2 It shall be possible to represent, using ASCII, the following data types:

- integer numbers
- floating-point numbers
- character strings
- enumerations (list of values, each represented by a symbol, i.e. a character string)

ascii-3 Units shall always be specified, if relevant, for integer and floating point data fields.

ascii-4 A given data field shall always use the same unit in all records of a given data set.

ascii-5 The primary choice of units should be SI units, except for:

- angles which should be represented using degrees (instead of the SI unit radians)
- temperatures which should be represented using Celsius degrees

See the following table.

Table 6.2.0–1 Earth Explorer - Primary Units

Data	Unit	ASCII Symbol
Length	meter	m
Mass	kilogram	kg
Time Duration	second	s
Angle	degree	deg
Frequency	hertz	Hz
Temperature	celsius	C
Amount of substance	mole	mol
Pressure	pascal	Pa

ascii-6 It is acceptable to use secondary units if this results in a clearer, more logical and/or smaller size (e.g. less bytes) representation. Acceptable secondary units are engineering units, i.e. orders of magnitude of 10^3, 10^6, 10^9... or 10^{-3}, 10^{-6}, 10^{-9}... from the SI units.

For example, milli-seconds (unit 10^{-3} s) or micro-seconds (unit 10^{-6} s) are acceptable, but NOT other odd sub-divisions such as tenth of milli-seconds (unit 10^{-4} s).

The ASCII symbol for those secondary units shall be those described in Table 6.2.0–1, preceded by "`10+n `" or "`10-n `" (where n is 3, 6, 9...), e.g. "`10-3 deg`" or "`10+6 Hz`"

ascii-7 Where secondary units are used (see ascii-6), it is acceptable to represent the unit with widely used symbols rather than the "`10+n `" or "`10-n `" form. See the following table for acceptable "widely used" symbols.

Table 6.2.0–2 Earth Explorer - Widely used Secondary Units

Data	Unit	ASCII Symbol
Length	kilo-meter meter milli-meter	km m mm
Mass	kilogram gram	kg g
Time Duration	second milli-second	s ms
Angle	degree	deg
Frequency	giga-hertz mega-hertz kilo-hertz hertz	GHz MHz kHz Hz
Temperature	celsius	C
Amount of substance	mole	mol
Pressure	pascal	Pa

ascii-8 Integer data fields shall be readable by software using a long integer number (32-bits).

ascii-9 Integer data fields should be represented using mantissa only if possible, for readability (i.e. 123000 rather than 123E+03). See also ascii-5 and ascii-6.

ascii-10 *(we need to define an hexadecimal representation for e.g. TM data, TBD)*

ascii-11 Floating-point data fields shall be readable by software using a double floating point number (64-bits).

ascii-12 Floating-point data fields should be represented using mantissa only if possible, for readability (i.e. 123.456 or 0.012, rather than 1.23456E+02 or 1.2E-02). See also ascii-5 and ascii-6.

ascii-13 Character string data fields shall be used to represent any text, without restrictions.

ascii-14 Enumeration data fields shall be readable by software using a character string. Note that the possible values are limited to a specific list of character strings, defined for each enumeration field.

ascii-15 Boolean data shall be represented using enumeration data fields, using the appropriate values (e.g. "TRUE" / "FALSE", "ON" / "OFF", "YES" / "NO", etc...).

ascii-16 Date and time fields shall be represented using character strings, using one of the standard ASCII time formats, as specified in [MCD], preferably a CCSDS format. Usage of a format containing the time reference (TAI, UTC, UT1, GPS...) should be systematic. Examples:

- UTC=2000-01-01T00:00:00 (CCSDS time, with UTC time reference)
- TAI=2004-07-04T12:34:56.987654 (CCSDS time, with TAI time reference)
- 20040704T123456 (compact CCSDS time, typically for usage within file names)

ascii-17 Angles shall be represented using numbers between -180 and +180 deg.

ascii-18 Geographical locations shall be represented using:

- latitude: from -90.0 deg (South Pole) to +90.0 deg (North Pole)
- longitude: from -180.0 deg (Greenwhich Meridian) to +180.0 deg (Greenwhich Meridian again)

6.3 Binary Data Representation

binary-1 Data shall be stored following the C Ansi data structure definition, with the exception of "long long" for 64 bit integers.

binary-2 The data types listed in Table 6.3.0–1 shall be used to store information:

Table 6.3.0–1 Binary Data types

variable type	C type	Range
character string	char unsigned char	{-128, +127} [0, +255]
1-byte integer	char unsigned char	{-128, +127} [0, +255]
2-bytes integer	short unsigned short	{-32 768, +32 767] [0, +65 535]
4-bytes integer	long unsigned long	[-2 147 483 648, +2 147 483 647] [0, +4 294 967 295]
8 -bytes integer	long long unsigned long long	[-9 223 372 0368 54 775 808, +9 223 372 036 854 775 807] [0, +18 446 744 073 709 551 615]
4-bytes single-precision floating-point	float	[-3.40282347e+38, 3.40282347e+38] [-1.17549435e-38, 1.17549435e-38]
8-bytes double-precision floating-point	double	[-1.79e+308, 1.79e+308] [-2.22e-308, 2.22e-308]

binary-3 Data structures smaller than a byte should NOT be used to store information (i.e not single bit and no bit sub-fields).

binary-4 A given data field shall always use the same unit in all records of a given data set.

binary-5 Allowed units are the same as described in ascii-5, ascii-6, ascii-7.

binary-6 Boolean data shall be stored using one of the integer types. The convention for values to be used shall be:

- 0: for e.g. “FALSE”, “OFF”, “NO”, etc...
- 1: for e.g. “TRUE”, “ON”, “YES”, etc...

binary-7 The bit/byte numbering convention is:

- bit 0 is the least significant bit
- byte 0 is the least significant byte

binary-8 Integer representation should be used rather than floating-point representation

binary-9 The IEEE 754 storage format for single-precision floating point numbers is represented in Figure 6.3.0–1. There are 3 fields:

- a 23-bit fraction, 0 being the least significant bit
- an 8-bit biased exponent
- a 1-bit sign (“s” in the figure)

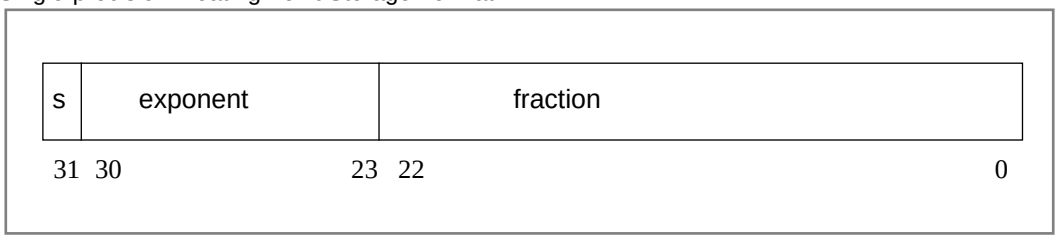
binary-10 The extreme absolute values which can be represented are:

- 3.40282347e+38 for the maximum
- 1.17549435e-38 for the minimum

binary-11 The number of significant decimal digits is between 6 and 9, that is, at least 6 digits, but not more than 9 digits are accurate.

.

Figure 6.3.0–1 Single-precision Floating Point Storage Format



binary-12 The IEEE 754 double-precision floating point storage format is represented in Figure 6.3.0–2. There are 3 fields:

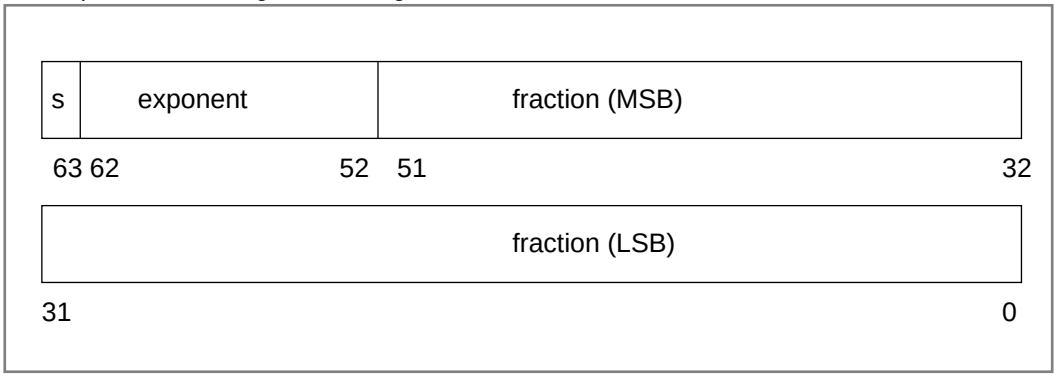
- a 52-bit fraction, 0 being the least significant bit.
- an 11-bit biased exponent
- a 1-bit sign (“s” in the figure)

binary-13 The extreme absolute values which can be represented are:

- 1.79e+308 for the maximum
- 2.22e-308 for the minimum

binary-14 The number of significant decimal digits is between 15 and 17, that is, at least 15 digits, but not more than 17 digits are accurate.

Figure 6.3.0–2 Double-precision Floating Point Storage Format



binary-15 The IEEE 754 convention for representing extreme numbers (infinity, zero, etc.) shall be used to store such values.

binary-16 Date and time fields shall be represented with integers, using one of the standard transport time formats, as specified in [MCD].

binary-17 Angles shall be represented using the same conventions as defined in ascii-17.

binary-18 Geographical locations shall be represented using the same conventions as defined in ascii-18.

binary-19 Data shall be stored in data files in the order from most-significant to least-significant byte (i.e. no byte-swapping as is used by default on some computer platforms).

binary-20 Data should be aligned on “4-bytes boundaries”, in order to avoid problems with structures on some computer platforms or with some compilers, i.e.:

- characters or 1-byte integers should always be grouped by 4
- 2-bytes integers should always be grouped by 2

7 Headers Content

7.1 Fixed Header

The Fixed Header is an XML structure. Many of its fields are redundant with the File Name elements, but are present in more readable form in the Fixed Header, whereas in File Name they are more compact for obvious reasons.

Table 7.1.0–1 Earth Explorer File - Fixed Header XML Structure

Tag Name	Type	Unit	Precision	C Format	Comment
File_Name	string			%s	It is a repetition of the Logical File Name, i.e. the File Names excluding the extension. This allows this field to be independent from the storage in 1 complete file or 2 separate files for Header and Data Block
File_Description	string			%s	A 1-line description of the File Type. Each Mission shall define the list of official file descriptions (per File Type). For Cryosat this is defined in [CS-ICD]. For GOCE this is defined in [GO-ICD].
Notes	string			%s	Multi-lines free text. This can be used for any type of comment, relevant that instance of the file.
Mission	string			%s	A 1-word description of the Mission, coherent with the Mission element in the File Name. See Section 4.1.1 for the official list.
File_Class	string				A 1-line description of the file class, coherent with the File Class element in the File Name. Each Mission shall define the list of official file classes. For Cryosat this is defined in [CS-ICD]. For GOCE this is defined in [GO-ICD].
File_Type	string			%10s	It is a repetition of the File Type element in the File Name.
Validity_Period	structure				see Table 7.1.0–2 for elements content
File_Version	integer			%04ld	It is a repetition of the File Version element in the File Name. Must start at 1 (not 0).
Source	structure				see Table 7.1.0–3 for elements content

Table 7.1.0–2 Fixed Header - Validity Period XML Structure

Tag Name	Type	Unit	Precision	C Format	Comment
Validity_Start	string			%23s	This is the UTC Validity Start Time, coherent with the Validity Start Time in the File Name, but in CCSDS ASCII format with time reference. Note that this can have the special value indicating “beginning of mission” (without an absolute time specified) as defined in [MCD] and Section 4.1.5.2. This format is defined in [MCD].
Validity_Stop	string			%23s	This is the UTC Validity Stop Time, coherent with the Validity Stop Time in the File Name, but in CCSDS ASCII format with time reference. Note that this can have the special value indicating “end of mission” (without an absolute time specified) as defined in [MCD] and Section 4.1.5.2. This format is defined in [MCD].

Table 7.1.0–3 Fixed Header - Source XML Structure

Tag Name	Type	Unit	Precision	C Format	Comment
System	string			%s	Name of the Ground Segment element creating the file (e.g. FOS, PDS, SSALTO...)
Creator	string			%s	Name of the tool, within the Ground Segment element, creating the file (e.g. CS-MCS, IPF1...)
Creator_Version	string			%s	Version of the tool (e.g. 1.0, 2.1a ...)
Creation_Date	string			%23s	This is the UTC Creation Date, in CCSDS ASCII format with time reference. This format is defined in [MCD].

Ex. 7.1.0–1 Earth Explorer Files - Fixed Header

```
<Fixed_Header>
        <File_Name>logical file name</File_Name>
        <File_Description>1-line file description</File_Description>
        <Notes>
                free text, free format
                several lines if needed
        </Notes>
        <Mission>mission name</Mission>      (e.g. Cryosat)
        <File_Class>1-line file class description</File_Class>
        <File_Type>TTTTTTTTT</File_Type>
        <Validity_Period>
                <Validity_Start>UTC=yyyy-mm-ddThh:mm:ss</Validity_Start>
                <Validity_Stop>UTC=yyyy-mm-ddThh:mm:ss</Validity_Stop>
        </Validity_Period>
        <File_Version>vvvv</File_Version>
        <Source>
                <System>name of system creating the file</System>
                <Creator>name of tool creating the file</Creator>
                <Creator_Version>version of tool</Creator_Version>
                <Creation_Date>UTC=yyyy-mm-ddThh:mm:ss</Creation_Date>
        </Source>
</Fixed_Header>
```

7.2 Variable Header

7.2.1 General Considerations

The Variable Header is specific to each File Type.

However, there are a number of desirable fields which should be present in the Variable Header. The presence of these fields, and their format, is to be defined on a case-by-case basis and recorded in the relevant ICD and/or file format description document.

The desirable fields:

- type of data block (XML, ASCII, binary)
- reference to a validation schema for the data block:
  - for XML data blocks, this is redundant with the reference in the data block XML
  - for non-XML data block, this is the only place to reference a validation schema
- name of input files used, if any, to generate the file
- other general contextual data for the file
- name of original non-XML data file (if file had to be renamed to respect the standard)
- reference of formal document(s) describing the data format and content

7.2.2 Variable Header Content for Binary Data Blocks

In addition to the above, in the case of binary data blocks the following fields are mandatory:

- byte ordering information, to allow compensation for byte-swapping on certain computer platforms

Table 7.2.2–1 Variable Header - Byte Ordering

Tag Name	Type	Unit	Precision	C Format	Comment
Byte_Order	string			%4s	either “3210” (from most- to least-significant byte) or “2301” (every 2 bytes are swapped) Byte 0 is the least significant byte, as defined in binary-7 the standard byte order is “3210”, as defined in binary-19

7.2.3 Variable Header Content for Binary Data Products

Binary Data Products produced by each Mission should have:

- a number of binary Data Sets
- each binary Data Set structure as a set of fixed-size and fixed-structure binary records

NOTE: ground processing software shall generate data sets with fixed-size and fixed-structure binary records. However, for Level-0 products, the size and structure of the binary records is directly related to the size and structure of the telemetry received from the satellite. In this case, it is possible that records do not have fixed size and/or structure.

It is strongly advised to negotiate with the space segment that each type of packet (uniquely identified by information in the packet header, e.g. APID) has fixed size and structure, as far as possible.

In addition to the file-level information, the Variable Header shall provide, for each binary Data Set, the following fields:

- offset from start of Data Block (in bytes)
- size (in bytes)
- bit and byte ordering information

The format of this information shall be defined consistently for all binary Data PRoducts within a given mission.

8 Data Block Contents

This is specific to each File Type.

9 Availability of Tools

A number of tools are provided by ESA to help validate, display, read and write Earth Explorer files. These tools are briefly introduced in the following sections, and their functionality and usage is fully described in [IO-SUM].

See Figure 9.6.0–1 for an overview diagram show ing usage of these tools.

9.1 File Validation

Earth Explorer files syntax, for what concerns the XML parts, is formally controlled using XML standard techniques. These involve:

- an XML schema for each File Type: the XML schema is a file describing formally the constraints to be applied to each file of the corresponding File Type; these constraints cover:
  - file structure (i.e. the sequence of tags)
  - tag values type and allowed range
- a central repository, under ESA control, where all XML schemas are stored, and made available through the Web
- validation tools, which analyze files and indicate syntax errors, by using the corresponding XML schema (accessed locally or through the Web)

The validation tools available, on a variety of computer platforms, are:

- a standalone executable tool
- optional validation features in reading/writing routines libraries, to be called from user applications, with interfaces for various computer languages

For what concerns non-XML parts, validation tools will also be available. These involve:

- a validation schema, in syntax TBC
- validation tools, described in TBC

9.2 File Display

XML files are, by nature, readable by humans, but the syntax makes them more cumbersome than strictly necessary.

There are, however, standard XML techniques to easily translate XML files in order to provide customized, highly user-friendly visualization of their contents. These involve:

- a set of XML style-sheets for each File Type: an XML style-sheet is a file describing formally the translation to be applied to each file of the corresponding File Type; these translations can be aimed at anything, in this case:
  - file visualization as HTML code, using any HTML tool such as a Web browser; web browser support more and more direct XML visualization, but a translation to HTML allows to provide a more user-friendly visualization (e.g. tables)
  - file visualization as simple ASCII text, using any text tool such as a text editor; data can be presented without the burden of the XML tags, in a "good old ASCII" style, which is not formal but readable
- a central repository, under ESA control, where all XML style-sheets are stored, and made available through the Web
- translation tools, which analyze files and translate them, by using the corresponding XML style-sheet (accessed locally or through the Web)

The translation tools available, on a variety of computer platforms, are:

- a standalone translation tool
- translation features available as standard features in XML-capable Web browsers

Note also that XML-capable Web browsers provide standard hierarchical visualization of XML files, which gives a level of readability intermediate between straight XML code and specialized, translated code.

9.3 File Conversion

The same translation techniques used for user-friendly display of XML files (as described in Section 9.2), can be used to translate XML files for any purpose.

XML style-sheets can therefore also be used for other goals, such as:

- conversion between versions of a File Type format definition (which is expected to evolve during development of the Earth Explorer ground segments); this will be particularly useful for test data files evolution, following file format evolution
- conversion to formats used by other systems
- conversion to formats which facilitate ingestion by post-procesing tools (e.g. data analysis tools such as IDL, PV-Wave, Excel...)

Note that the translation techniques based on XML style sheets allow advanced manipulations, such as:

- hiding parts of file contents (e.g. to remove irrelevant/superfluous information)
- modify contents order (e.g. to transform a list of identically structured records into a row-column-based table, if relevant)
- adding any arbitrary contents (e.g. to add "row/column headers" to a table)
- adding new contents based on combination of existing file contents (e.g. to add a "totals row" at the bottom of a table)

Translation from other formats into Earth Explorer formats are not feasible using standard techniques. It requires ad-hoc tools to be implemented. Note, however, that the output part (to generate the Earth Explorer version of the converted file) can make use of the writing routines described in Section 9.4.

9.4 File Reading and Writing

Libraries of routines with APIs for reading and writing Earth Explorer files is available, for a variety of computer platforms. These routines can be called from user applications, and have interfaces for various computer languages.

Typical functions provided are:

- read value of XML element(s), given the XML tag name and a file
- read file name elements, given the file name string
- read Fixed Header elements, given a file
- etc...

The functionality of these libraries includes (not exhaustive):

- reading of Earth Explorer file headers, transparently (i.e. without a-priori knowledge of headers structure)
- reading the value of a tag, given its tag name and type (this requires a-priori knowledge of the file structure)
- reading of all values of a given tag, given its tag name and type (this requires a-priori knowledge of the file structure)
- reading a complete file XML structure in memory
- incrementally create a complete file XML structure in memory
- read value of a tag from the structure in memory
- modify the value of a tag from the structure in memory
- modify complete header contents in the structure in memory
- write the complete XML structure from memory to file

A “safe edifor” for interactively reading and writing XML-based Earth Explorer files is also available, for a variety of computer platforms. This is a standalone executable tool, with the following functionality:

- interactive visualizing and editing of any XML-based Earth Explorer file, using WYSIWIG HMI techniques
- control of data types and allowed ranges through use of the relevant XML schemas

9.5 Header and Data Block Extraction

To allow usage of the tools described in the previous sections, and due to the non-XML nature of the Data Block in some Earth Explorer files (see Section 5.2.3.2 and Section 5.2.3.3), a tool to extract separately the Header and Data Block of any Earth Explorer file is available.

This tool is available for a variety of computer platforms.

The tool respects the file naming conventions described in Section 4.2.2.

9.6 Header and Data Block Packaging / Un-Packaging

To support packaging and un-packaging of Header and Data Block files, and optional compression and decompression, standard tools (not developed specially for Earth Ecplorer missions) are available:

- zip
- tar
- gzip / gunzip

These tools, originally developped for Unix platforms, are available for a variety of computer platforms:

- supported by “winzip” for Windows
- supported by “stuffit” for MacOS

Figure 9.6.0–1 XML Tools Usage Overview