estec

European Space Researchand Technology Centre

Keplerlaan 12201 AZ NoordwijkThe Netherlands
Tel. (31) 71 5656565Fax (31) 71 5656040

DOCUMENT

Earth Observation

Ground Segment

File format Standard

PE-TN-ESA-GS-0001 EO GS File Format Standard 3.0.1_b.docx

Prepared by	**System Support Division EOP-PE**
Reference	**PE-TN-ESA-GS-0001**
Issue	**3**
Revision	**0.1**
Date of Issue	**01 Nov 2023**
Status	**Authorised**
Document Type	**TN**
Distribution	

European Space Agency
Agence spatiale européenne

APPROVAL

Title	
Issue 3	**Revision** 0.1
Prepared by	**Date**
System Support Division EOP-PE	01 Nov 2023
Approved by	**Date**
Pierre Viau EOP-PE	

CHANGE RECORD

Issue	Date	Page	Section	Req't	Description of Change
0.5	3-10-2001	All			- initial draft Issue, limited distribution
0.6	30-10-2001	All			- draft issue, distributed for comments
0.9	9-11-2001	All			- draft issue, for Ground Segment Requ. Review
1.0	1-05-2002	All			- updated after comments from all reviewers
1.1	24-05-2002	All			- clarified file naming - consolidated Fixed Header content - added binary records description in XML Variable Header
1.2	12-07-2002	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	18-10-2002	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	13-06-2003	All	All 5.2 7.2.3		- small clarifications (see change bars) - added validation schema reference for non-XMLdata - removed record structure description from VariableHeader
2.0	3-05-2012	All			- in title and text, obvious changes from "Earth Explorer" to "Earth Observation" (NB: no change bars for those) - formal introduction of the concept of file format standard tailoring - file naming upgrade (3 alphanumeric letters for mission ID, extension changed to EOF) - removal of the formal list of mission IDs (which is a tailoring issue) - relaxing of the file name size constraint - clarification of 4-characters File Category usage - clarification of other ambiguous or obsolete text - correction of XML schema usage - correction of minor typos
3.0	18-04-2015	All	All		- Updated XML tag for the complete header from "<Earth_Explorer_Header>" to "<Earth_Observation_Header> - Updated XML tag for the complete file has from "<Earth_Explorer_File>" to "<Earth_Observation_File>" - Several Data Block files allowed in packaged files. - Clarification of and relaxing definition of the extension used for separated Data Block files

					- Clarified packaging options: tar , gzip removed from standard. Zip retained. - Clarification on File Instance ID - Clarification on Version Number - Clarification on Data Block Syntax - Clarification on Binary Data Representation - Clarification on Hexadecimal Data representation in ASCII. - Clarification on Leap Second representation - EOFFS_Version tag added in Fixed Header. - Variable Header content amended for binary files and data products - Clarification on File Validation. - Editorial corrections.
3.0.1	01-11-2023	All	All		- Update of the internal structure of the file type name to reflect PDGS standard.

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 Observation Missions within the Agency’s Earth Observation Programme.
In order to promote a common approach for Ground Segment data exchanges within those missions, ESA has defined a file format standard, recorded in this document.
The standard was originally defined for the Earth Explorer missions, and is now extended to support all current and future Earth Observation missions, which results in:

- a document title change , since Version 2.0.
- (very) few amendments to the standard, which only concern file naming, since Version 2.0
- few amendments to the standard that concern file content ,XML Tags and extensions since Version 3.0
- Standard has subsequently be amended considering the PDGS standard baseline (v 3.0.1)

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 Observation Missions.
The expected benefits of the common use of this standard are:

- increasing standardization of formats over several missions
- use of industry and commercial 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 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 Observation Missions ground segments.

Note that a number of options are described in this document, for several aspects of the file formats, allowing for flexibility. This is, for example, relevant for File Type naming, Variable Header structure,
Data Block structure... (these terms are fully described in the following sections). For a given mission, the implementation of those options is typically described by each mission in a “File Format Standard Tailoring” document.
The present standard does not address higher-level formatting and packaging needed, for example, for long term data preservation (e.g. SAFE directories) however it shall be applied to the data files contained within these higher level packages.
All documents formally describing file format details shall take into account this standard, and the tailoring document for the relevant mission. These documents are typically Interface Control Documents and File Format Specifications Documents.

1.4 Acronyms and Terminology

1.4.1 Acronyms

ANSI	American National Standards Institute
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BUFR	Binary Universal Form for the Representation of meteorological data
DBL	Data Block
DFDL	Data Format Definition Language.
DORIS	Doppler Orbitography and Radio-Positioning Integrated by Satellite
EOP	Earth Observation Programme
ESA	European Space Agency
ESL	Expert Support Laboratory
FITS	Flexible Image Transport System
FH	Fixed Header
FOS	Flight Operations Segment
GPS	Global Positioning System
GSV	Ground Segment Validation
JPEG	Joint Photographic Expert Group - Graphical File Format
HDF5	Hierarchical Data Format#5
HK/TM	House-Keeping TeleMetry
HMI	Human-Machine Interface
ICD	Interface Control Document
ISP	Instrument Source Packet
MMI	Man-Machine Interface, see HMI
NetCDF	Network Common Data Form – Auto-Documented Data Format
LTA	Long-Term Archive
OAIS	Open Archival Information System
PDS	Payload Data Segment
PNG	Portable Network Graphics - Graphical File Format
POD	Precise Orbit Determination
S1/2/3	Sentinel-1/2/3
SAFE	ESA Standard Archive Format for Europe
SDE	Software Development Environment
SIP	Submission Information Package (OAIS)
TBC	To Be Confirmed

TBD	To Be Defined
TIFF	Tag(ged) Image File Format - Graphical File Format
USF	User Services Facility
VH	Variable Header
Web	See WWW
WWW	World-Wide-Web
W3C	World-Wide-Web Consortium (http://www.w3.org)
WYSIWYG	“What You See Is What You Get”
XLS	Excel File (extension)
XML	eXtensible Markup Language
XSL	XML Style Sheet Language
XLST	XSL Transformation

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. A logical file can contain one or more Data Block(s).
- 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
- XML Schema — Schemas are used to describe the structure and syntax of XML files and, as a non-W3C extension, of binary files as well.
- XML Vocabulary — A set of XML tags used to describe a particular data structure. A vocabulary is defined by an XML Schema. Various XML vocabulary have been developed e.g. eXtensible HyperText Markup Language (XHTML)

- DFDL — Data Format Definition Language, an XML Schema-based standard used to describe the physical content of a binary file in term of format description and parameter range/value.

1.4.3 Document Conventions Document

This document uses specific fonts in the following cases:

courier_bold_italic	for file names, or elements of file name; these are not actual character strings, 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 or are relevant to give context.

[MCD]	Earth Observation Mission CFI - Conventions Document EO-MA-DMS-GS-0001
[GEN-SUM]	Earth Observation Mission CFI - General SUM EO-MA-DMS-GS-0002
[IO-SUM]	Earth Observation Mission CFI - Data/File Handling SUM CS-MA-DMS-GS-0007/008
[XML]	XML Reference: http://www.w3.org/TR/REC-xml
[SCHEMA]	Handbook for Earth Observation XML and Binary Schemas PE-TN-ESA-GS-121
[IEEE]	IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std 754-1985 (IEEE 754) Institute of Electrical and Electronics Engineers, 1985
[DFDL]	DFDL Specification: https://www.ogf.org/ogf/doku.php/standards/dfdl/dfdl
[SREC]	Motorola S-Record Format: http://en.wikipedia.org/wiki/SREC_(file_format)
[CFIFS]	Earth Observation Mission Software File Format Specification, PE-ID-ESA-GS-584

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
- one or more Data Block(s)

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

- a single Physical File
    1. one single Physical File constituting of a Header and a(single) XML Data Block.
- two or more separate Physical Files:
    1. a Header File
    2. one or more Data Block File(s)

The advantage of separating Header and Data Block(s) 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(s): scientific data

Logical and physical files should be kept as simple as possible:

- if possible, only 1 Data Block shall be used, and in that case:
    1. Header and Data Block should be structured into single Physical File only if both Header and Data Block are of XML form.
    2. if the Data Block contains binary data, Header and Data Block should be structured into separated Physical Files.
- in the case of multiple Data Blocks, Header and Data Blocks should be structured into separated Physical Files.

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 should structure the VH accordingly in a modular way, for example, the VH of scientific data products should consist of a common product header and a specific product header.

3.3 Data Block

The Data Block format can either be

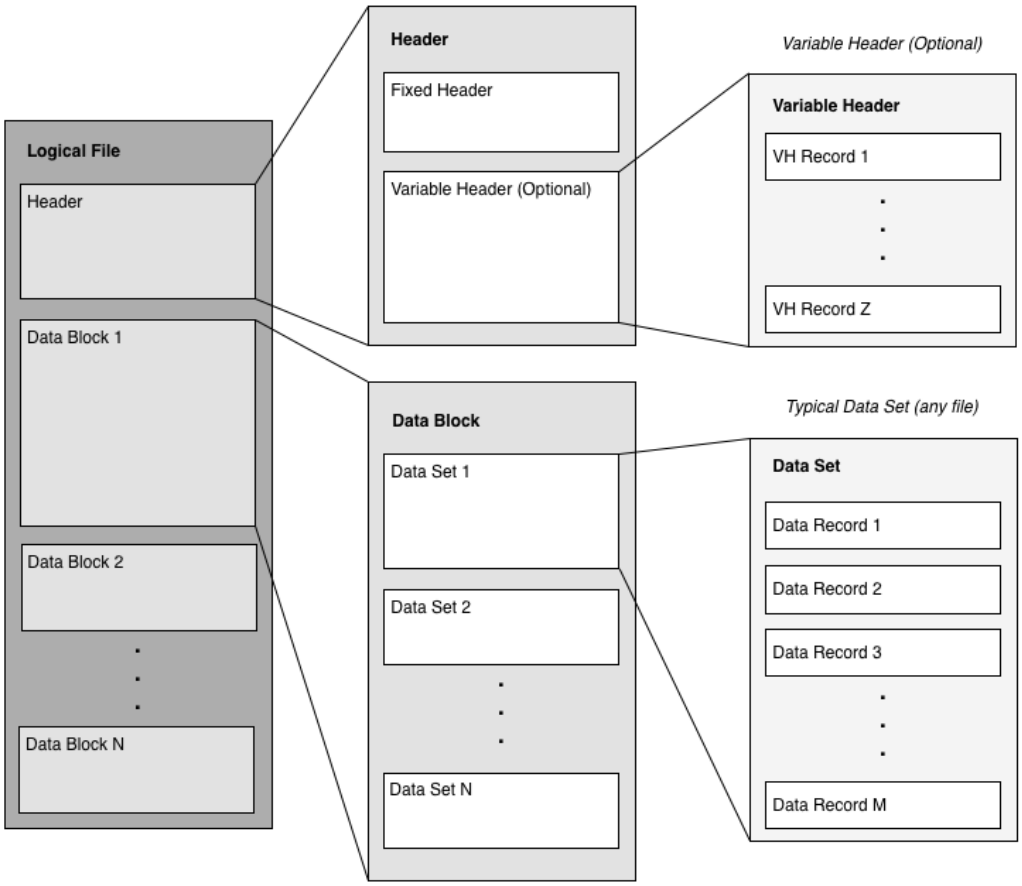
- user-defined or
- compliant with standard/commercial formats. See also Section 4.2.3 for Physical Data Block File extension.

In the case of user-defined 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 if possible.

NOTE:
This describes the general case for user-defined Data Blocks. There can be exceptions to the Data Block structures (such as non XML ASCII and Binary Data Block) or the Data Block can be compliant with a standard/commercial format.
(See Section 5.2.3 and Section 8).

Figure 3.3-1 Earth Observation 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, USB storage, electronic links,...):

- file compression may be used for data transfer or media space usage efficiency

- if the Header and the Data Block(s) are stored as separate Physical Files, it is vital that all these files are packaged and distributed together.

The allowed mechanisms used are:

- packaging and compression using the “zip” format

NOTES:

- If other legacy packaging mechanism deviating from this standard are used, these shall be defined in the each Mission Specific Format standard tailoring
- the packaging has also the advantage that it could also be used to distribute, 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).
- The present standard does not address the higher level re-packaging of files for specific applications (e.g. long-term archival, preservation, storage or dissemination) like SAFE and OAIS SIP do.
- Even when data is formatted according to different standard (e.g. as a SAFE directory) it is **strongly recommended** that all transport/distribution is performed with the data/directory packaged into a single file according to this standard i.e. using .zip.

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. 4.1-1 Earth Observation Logical File Name

`MMM_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 Observation ground segments.
The file name elements are described in the following table.

Table 4.1-2 Earth Observation Logical File Name Elements

Naming Element	Description	Comment
MMM	Mission ID, for example: - ECR for EarthCare mission - S1A for Sentinel-1 mission, satellite A - S2_ for Sentinel-2 mission, all satellites	3 uppercase letters, can contain digits and underscores "_".
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 - L0__NAVATT: Navigation data packet 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 on the File Category (see below). Possible examples include: yyyymmddThhmmss_YYYYMMDDTHHMMSS_vvvv (Validity Period and Version Number) YYYYMMDDTHHMMSS_oooooooo	Maximum 64 characters, can contain uppercase letters, digits and underscores

	(Creation Date and Orbit Number)	"_".[2]

1. Note that earlier (Earth Explorer) missions, applying a version of the standard earlier than Version 2.0, use 2 letters for the Mission ID.
2. Note that earlier (Earth Explorer) missions, applying a version of the standard earlier than Version 2.0, use a maximum of 41 characters for the Instance ID. The difference comes from the above Mission ID change (from 2 to 3 characters).

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

MMM_CCCC_TTTTTTTTT_instanceID

Consists of 3 characters, either uppercase letters or digits or underscores "_"
Note that earlier (Earth Explorer) missions, applying a version of the standard earlier than Version 2.0, use 2 letters for the Mission ID.
Each Mission shall define the Mission ID in its file format standard tailoring document.
A typical Mission ID could be:
• ECR for the EarthCare mission
• S1A for the Sentinel-1 mission, files relevant for satellite A
• S2_ for the Sentinel-2 mission, files relevant for all satellites

4.1.2 File Class

MMM_CCCC_TTTTTTTTT_instanceID

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 independent 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
- OFFL for data products generated Off-line (Reprocessing)
- TEST for internal tests
- others TBD

Each Mission shall define the list of authorized File Classes in its file format standard tailoring document.

4.1.3 File Type

MMM_CCCC_TTTTTTTTTT_instanceID

Consists of 10 characters, (identified as TTTTTTTTTT above) 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 in its file format standard tailoring document.
The following sub-sections describe the File Category, a vital part of the File Type, and the standard for instrument data products File Types.

NOTE:
The management of format evolution and its versioning is performed transparently by means of versioned schemas according to [SCHEMA] for both the XML part as well as for any User-defined binary data block and is not propagated at filename level.
The format versioning for Standard/Commercial binary formats (e.g. different version of JPG, NetCDF, PDF, etc) is performed in the Variable Header (see section 7.2)

The recommended split of the File type element is:

TTTTTTTTTT = FFFDDDDDDD
Where:
- FFF: is the "File Category "
- DDDDDDD: is the "Semantic Descriptor"

4.1.3.1 File Category

Within the File Type, to organize the files properly, the initial 3 characters "FFF" shall be reserved for a File Category sub-element.

Typical File Categories could be:
- MPL for mission planning files
- TLM for telemetry retrieval files
- AUX for auxiliary data files
- CAL for Calibration files
- III for data products files, where III is product-specific (see Section 4.1.3.3)
- others TBD

Each Mission shall define the list of authorized File Categories in its file format standard tailoring document.

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 Semantic Descriptor

Within the File Type, to organize the files properly, the last 7 characters "DDDDDDD" shall be reserved for a Semantic Descriptor sub-element providing the description of the content of the product. To improve readability, if possible, the first character should be an underscore. However, if necessary, all 7 alphanumeric characters could be used.
Each Mission shall define the list of authorized Semantic Descriptors in its file format standard tailoring document.

4.1.3.3 Data Products: File Category and File Type

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

For data products, the File Type should be structured as FFFDDDDDDD, where:

- FFF is the File Category, which in Data Products typically defines the *product level*. Each Mission PDGS shall define the list of authorized strings in its file format standard tailoring document.

  Typical data products File Categories could be:

  - RAW for raw binary CCSDS data
  - L0_ for Level-0
  - L1B for Level-1B
  - ANC for ancillary data

- DDDDDDD is the Semantic Descriptor. To improve readability, if possible, the first character should be an underscore. However, if necessary, all 7 alphanumeric characters could be used.
  Each Mission PDGS shall define the list of authorized strings in its file format standard tailoring document, and it is highly desirable that is there defined also a further, semantically meaningful, sub-structuring of such Semantic Descriptor, e.g. IIIXXXX where "III" is instrument specific and "XXXX" provides mode/semantic description or IIIIXXX where "IIII" is instrument specific and "XXX" provides mode/semantic description.

  Typical data products Semantic Descriptor could be:

1. _XS_HR1 for FLEX RAW TM from HR1 sensor
2. OBS____ for FLEX Earth Observation products
3. _LR_VAU for FLEX LR sensor VAU TM

4. ESR_LND for TRUTHS Earth-Reflected Spectral Radiance observation over Land
5. HS_DRK1 for TRUTHS HIS instrument Dark calibration 1
6. HS_MOON for TRUTHS Lunar irradiance measurement

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

- RAW_XS_HR1 for FLEX HR1 Data Channel
- L0_OBS____ for FLEX FLORIS Instrument Measurement Level-0
- RWS_LR_VAU for FLEX VAU_TM for LR sensor Complete RAW slice
- L1BESR_LND for TRUTHS Earth-Reflected Spectral Radiance observation over Land Level-1b
- L0_HS_MOON for TRUTHS Lunar irradiance measurement Level-0

4.1.4 File Instance ID

MMM_CCCC_TTTTTTTTTT_instanceID

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 meaningful naming structures yielding unique file names.
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”. This does not apply to file of category “AUX_” (see Section 4.1.3.1) that might require different file instance ID shapes. 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 in its file format standard tailoring document.

NOTES:
File Instance ID “shapes” should have a maximum of 40 characters, to respect File Name size constraints (see Section 4.2.5).

Note that earlier (Earth Explorer) missions, applying a version of the standard earlier than Version 2.0, use a maximum of 41 characters for the Instance ID. The difference comes from the above Mission ID change (from 2 to 3 characters).

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):
- **yyyymmddThhmmss_YYYYMMDDTHHMMSS_vvvv** (i.e. Validity Period and Version)

- *YYYYMMDDTHHMMSS_ooooooo* (i.e. Creation Date and Orbit Number)
- *yyyymmddThhmmss_yyyymmddThhmmss_yyyymmddThhmmss_CCC_LLL_BB* (i.e. sensing period, creation date, Cycle number, Relative Orbit number, PDGS Processing Baseline)

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

If a Mission uses a Validity Period in file names, it shall define, for each File Type using it, the exact definition of the Validity Period in its file format standard tailoring document.

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]):
  1. "*00000000T000000*" for beginning of mission
  2. "*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]

- Leap Second representation is supported by making use of the “second” field be set to 60.

4.1.5.3 Version Number

Consists of N characters, all digits.

A new version number should only be needed when all preceding 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.
- In case of files are generated by independent entities whereby a sequential counter cannot be ensured, it is allowed to waive the sequentially requirement as long as uniqueness of the version number is ensured e.g. by allocation counter ranges to different entities.

4.1.6 File Name Sizes

This is specified in Section 4.2.5 (since it is a physical file name issue).

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. 4.2-1Earth Observation Physical File Name

MMM_CCCC_TTTTTTTTTT_*<instance ID>.<Ext>*

where <Ext> is a different extension depending on the physical file. See Section 4.2.2 , 4.2.3and Section 4.2.4 for possible extensions.

4.2.2 Single File vs Header and Data Block Files

The following Physical File Names shall be used:

- MMM_cccc_TTTTTTTTTT_*<instance ID>*.EOF [1]
  for a complete file (containg both the XML Header as defined in the ".HDR" file and the XML Data Block)
- **MMM_cccc_TTTTTTTTTT_*<instance ID>*.HDR**
  for Header file
- **MMM_cccc_TTTTTTTTTT_*<instance ID>*.<DBL_Ext>**
  for a Data Block file. (see Section 4.2.3 for details on the possible values of <DBL_Ext>)

NOTE:
In the case of multiple Data Block files, the Variable Header shall provide a descriptive list of the Data Block files(s). See Section 7.2.5.

4.2.3 Data Block Files Extension

The Physical Filename of a Data Block File always contains a dedicated extension <DBL_Ext>, as introduced in Section 4.2.2
The values of the Data Block File(s) extension <DBL_Ext> is to be defined on a cases by case basis and shall comply to the following conventions.

1. In order to avoid confusion with non Data Block files, the following values shall Not be used for <DBL_Ext>
   - "HDR" - reserved for Header Files
   - "EOF" - reserved for EOF single Files
   - "ZIP" – reserved for Package Files
2. This historical default value of <DBL_Ext> is "DBL".

[1] Note that earlier (Earth Explorer) missions, applying an earlier version of the standard, use the "**.EEF**" extension for complete files (containing both Header and Data Block).

3. However, the usage of standard/commercial format (e.g. XML, ASCII, XLS, PDF, BUFR, NetCDF, HDF5, JPEG, PDF, TIFF, FITS) and associated extensions (to a maximum size of 4 alphanumerical characters) instead of the historical default value “DBL” is permitted. This allows commercial reading software to directly open the Data Block file.eg
   MMM_cccc_TTTTTTTTTT_<*instance ID*>.XML
   MMM_cccc_TTTTTTTTTT_<*instance ID*>.TXT
   MMM_cccc_TTTTTTTTTT_<*instance ID*>.XLS
   MMM_cccc_TTTTTTTTTT_<*instance ID*>.PDF
   MMM_cccc_TTTTTTTTTT_<*instance ID*>.NC
   MMM_cccc_TTTTTTTTTT_<*instance ID*>.HDF5
   MMM_cccc_TTTTTTTTTT_<*instance ID*>.JPG
   MMM_cccc_TTTTTTTTTT_<*instance ID*>.TIFF
   MMM_cccc_TTTTTTTTTT_<*instance ID*>.NC

4. In the case of multiple Data Block files with identical standard/commercial format extensions, Data Block Files will be differentiated by adding an intermediate fixed sixed extension of 4 alphanumerical ([A-Z] [0-9]) characters eg
   MMM_cccc_TTTTTTTTTT_<*instance ID*>.R001.TIFF
   MMM_cccc_TTTTTTTTTT_<*instance ID*>.R002.TIFF
   MMM_cccc_TTTTTTTTTT_<*instance ID*>.R003.TIFF

4.2.4 Packaging and Distribution of Files

To support packaged files, the “zip” utility shall be used, as
- it compresses individual files before packaging
- it allows quicker retrieval of individual files within a compressed package

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

MMM_cccc_TTTTTTTTTT_<*instance ID*>.ZIP

for a “zip” package containing either:
1. a complete file (“.EOF”) after compression using “zip”
2. several files (typically “`.HDR`” and one or more “`.<DBL_Ext>`” after compression using “zip”
   See Section 4.2.3 for possible Data Block File extension `<DBL_Ext>`

Note that in both cases, the package can also contain supporting files (see notes below)

NOTES:
The packaging has also the advantage that it could also be used to distribute, 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.

4.2.5 File Name Sizes

The size of File Name depends on the size of the File Instance ID (see Section 4.1.4) and the size of the File Extension (see Section 4.2.1) but all Logical File Names should be smaller than 60 characters.
This minimizes the problems of file name truncation on legacy operating and file systems.
If a Mission violates this constraint for some file names, it shall clearly mention this deviation in its file format standard tailoring document.

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 the Header and the DataBlock(s), 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_Ext> See Section 4.2.3 for possible Data Block File extension.

- if the sender also creates the Header, as a separate file, the Header file shall use the Header name style, *file_name*.HDR, and the Header file and the Data Block file shall be sent together in a package as *file_name*.ZIP.
- if the sender creates the Header and merges the Header and the Data Block, the file shall use the complete file name *style, file_name*.EOF
- optional compression can be used before sending the single file as *file_name*.ZIP.

5.1.3 Earth Observation 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 Observation 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 representing values enclosed within tags. Tags can have one or more optional attributes. The syntax is shown below.

Ex. 5.1-1 Basic XML Tag Syntax

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

A few practical examples are given below.

Ex. 5.1-2 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 Observation 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 Observation 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)

Note that the “endianess” specification for a user-defined binary Data Block is provided by a specific “Byte-Order” field in the Variable Header. See Sections 7.2.2 and 7.2.5.

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 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 Observation 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_names 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_Names>)
NOTE: please notice the use of “s” (or the valid plural e.g. "List_of_Volcanoes" or "List_of_Dummies") after *Element_name* in List_of_Element_names.

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 structural flexibility, it shall be possible to use variable structure, by using the W3C **<xs:choice>** construct within the associated XML schema e.g.:

```
<xs:element name="Data_Take">
  <xs:complexType>
    <xs:choice>
      <xs:element name="Measurement" type="MeasurementType"/>
      <xs:element name="Calibration" type="CalibrationType"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

xml-15 Variable structure should be avoided as far as possible

xml-16 Although elements at different locations in a hierarchical structure (e.g. Medium_Level_Tag_1 and Low_Level_Tag_3 in Ex. 5.1-3) can formally have the same tag name according to the XML specification, this should be avoided since it can create confusion.

Ex. 5.1-4 Earth Observation 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*.EOF:

Ex. 5.2-1 Earth Observation File - Complete File Syntax

```
<?xml version="1.0" ?>
<Earth_Observation_File Validation-Schema-Reference>
     Header (same as the Header file file_name.HDR)
     Data Block
</Earth_Observation_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 the Header and Data Block File are always transferred together as a package (see Section 4.2.4)
- the XML Prolog is used to control XML specification version; it shall be used for every XML file, including ".EOF", ".HDR" and ".<DBL_Ext>" if relevant. See Section 4.2.3 for possible Data Block file extension.
- a reference to the XML schema reference (see [SCHEMA] for the recommended syntax) allowing to validate the file syntax shall always be provided as attribute to the first tag; see Section 9.1 for XML validation tool details
- the XML tag for the complete file has been updated from "<Earth_Explorer_File>" as for the Version 2.0 of the file format standard to "<Earth_Observation_File>"

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, and include the validation schema reference in the first tag):

Ex. 5.2-2 Earth Observation File - Header Syntax

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

```
        </Variable_Header>
    </Earth_Observation_Header>
```

NOTE:

- the XML tag for the complete header has been updated from "<Earth_Explorer_Header>" as for the Version 2.0 of the file format standard to "<Earth_Observation_Header>

5.2.3 Data Block Syntax

5.2.3.1 User-defined XML ASCII Data Block Syntax

Ex. 5.2-3 Earth Observation 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_Ext> (except that the Data Block shall add the XML Prolog as initial line, and include the validation schema reference in the first tag). See Section 4.2.3 for possible Data Block File extension <DBL_Ext>.

5.2.3.2 User-defined Non-XML ASCII Data Block Syntax

Ex. 5.2-4 Earth Observation File - Non-XML ASCII Data Block Syntax

```
<Data_Block type="ascii" Validation-Schema-Reference>
     <![CDATA[            (syntax TBC)
     Data Block Contents  (same as the Data Block file file_name.<DBL_Ext>)
     ]]>                  (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_Ext>.See Section 4.2.3 for possible Data Block File extension <DBL_Ext>.

5.2.3.3 User-defined Binary Data Block Syntax

In the case of user-defined binary format is chosen for the Data Block, the Data Block contents shall always be stored as a separate Data Block file (*file_name*.<DBL_Ext>). This is an exception to the general file syntax as shown in 5.2.1. In this case it is recommended to define a DFDL schema describing the format.
See Section 4.2.3 for possible Data Block File extension <DBL_Ext>.

NOTE:
Standard XML tools will not be able to parse and display the Data Block, and ad- hoc tools must be used.
The separation of Header from Data Block(s) avoids that failure in reading the Data Block prevents from reading the Header with XML tools, it also ensure that file handling (inventory, circulation, dissemination) can be performed based only on this XML Header. This allows accepting any binary formats for the Data Block part without the need to develop specific code and integrate specific libraries for each file_type

5.2.3.4 Standard/Commercial Binary Data Block Syntax

In the case of standard/commercial binary format is chosen for the Data Block, the Data Block contents shall always be stored as a separate Data Block file (*file_name*.XXX). , where XXX is the relevant Data Block file extension alternative.. This is an exception to the general file syntax as shown in5.2.1, see example in Ex. 5.2-1. See Section 4.2.3 for possible Data Block File extension.

NOTE:
Standard XML tools will not be able to parse and display the Data Block, and ad- hoc tools must be used.
The separation of Header from Data Block(s) avoids that failure in reading the Data Block prevents from reading the Header with XML tools, it also ensure that file handling (inventory, circulation, dissemination) can be performed based only on this XML Header. This allows accepting any binary formats for the Data Block part without the need to develop specific code and integrate specific libraries for each file_type

5.2.4 XML ASCII Data Set Syntax

Ex. 5.2-5 Earth Observation 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. 5.1-4), 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 Observation 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 Earth Observation File - XML ASCII File Syntax - Complete File ("`.EOF`")

```
<?xml version="1.0" ?>
<Earth_Observation_File Validation-Schema-Reference>
     <Earth_Observation_Header>
          <Fixed_Header>
               Fixed Header contents
          </Fixed_Header>
          <Variable_Header>
                Variable Header contents
          </Variable_Header>
     </Earth_Observation_Header>
     <Data_Block type="xml">
          <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_Observation_File>
```

Please note the XML Prolog as initial line, and the validation schema reference in the first tag.

Ex. 5.3-2 Earth Observation File - XML ASCII File Syntax - Header & Data BlockFiles (".HDR" & ".<DBL_Ext>")

Header File (file_name.HDR):

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

Data Block File (file_name.<DBL_Ext>):

```
<?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(s) contents are stored as separate files, they shall be distributed as a package with *file_name*.ZIP,.
For both files, note the XML Prolog as initial line, and the validation schema reference in the first tag.

5.3.2 Non-XML ASCII File Syntax

Ex. 5.3-3 Earth Observation File - Non-XML ASCII File Syntax - Complete File (".EOF")

```
<?xml version="1.0" ?>
<Earth_Observation_File Validation-Schema-Reference>
     <Earth_Observation_Header>
          <Fixed_Header>
               Fixed Header contents
               </Fixed_Header>
          <Variable_Header>
               Variable Header contents
          </Variable_Header>
     </Earth_Observation_Header>
     <Data_Block type="ascii">
          <![CDATA[           (syntax TBC)
          Data Block Contents (same as file file_name.<DBL_Ext>)
          ]]>                 (syntax TBC)
</Data_Block>
</Earth_Observation_File>
```

Ex. 5.3-4 Earth Observation File - Non-XML ASCII File Syntax - Header & Data Block Files (".HDR" & ".<DBL_Ext>")

Header File (file_name.HDR):

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

Data Block File (file_name.<DBL_Ext>*):* *ad-hoc ASCII syntax*

Note that if the Header and the Data Block contents are stored as separate files, they shall be distributed as a package with *file_name*.ZIP

5.3.3 Binary File Syntax

Ex. 5.3-5 Earth Observation File - Binary File Syntax

Header File (file_name.HDR):

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

Data Block File (file_name.<DBL_Ext>*):* *ad-hoc binary syntax*

Note that if Header and Data Block(s) contents are stored as separate files, they shall be distributed as a package with *file_name*.ZIP

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. Each mission shall clearly define the list of units to be used for each parameter field in its specific file format definition document.

ascii-5 The primary choice of units should be SI units (See the following table), except for:

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

Table 6.2-1 Earth Observation - 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 6.2-1, 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 6.2-2 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-herz mega-hertz kilo-herz 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 Hexadecimal data should be represented using a single convention in a specific mission (e.g. the Motorola S-record format, see [SREC]). The selected hexadecimal data representation shall be defined in the mission specific File Format Standard tailoring.

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)

Note : Leap Second representation is supported by making use of the “second” field be set to 60

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 (Date Line) to +180.0 deg (Date Line again)

6.3 Binary Data Representation

6.3.1 User-defined Binary Data Representation

Data shall be stored following the C ANSI data structure definition, with the exception of “long long” for 64 bit integers.

The data types listed in Table 6.3-1 shall be used to store information:

Table 6.3-1 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]

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

Note: This is not always possible such as in the case of satellite telemetry data or for data size efficiency reasons.

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

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

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...

The bit/byte numbering convention is:

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

Integer representation should be used rather than floating-point representation

The IEEE 754 storage format for single-precision floating point numbers is represented in Figure 6.3-2. There are 3 fields:

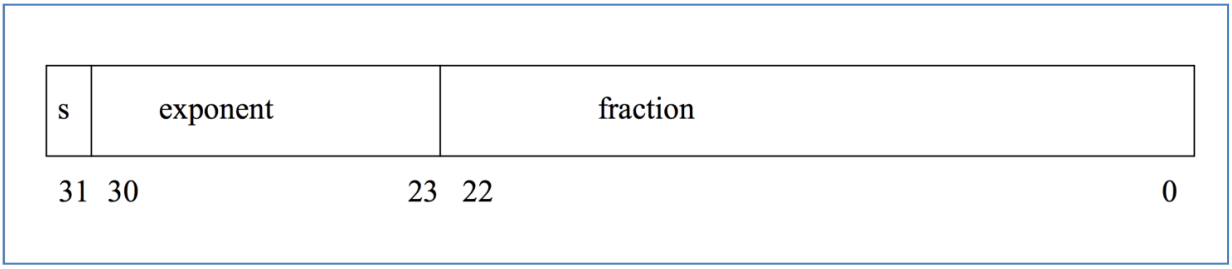
- a 23-bit fraction, 0 being the least significant bit
- an 8-bit biased exponent
- a 1-bit sign (“s” in the figure)

The extreme absolute values which can be represented are:

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

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 6.3-2 Single-precision Floating Point Storage Format



The IEEE 754 double-precision floating point storage format is represented in Figure 6.3-3. There are 3 fields:

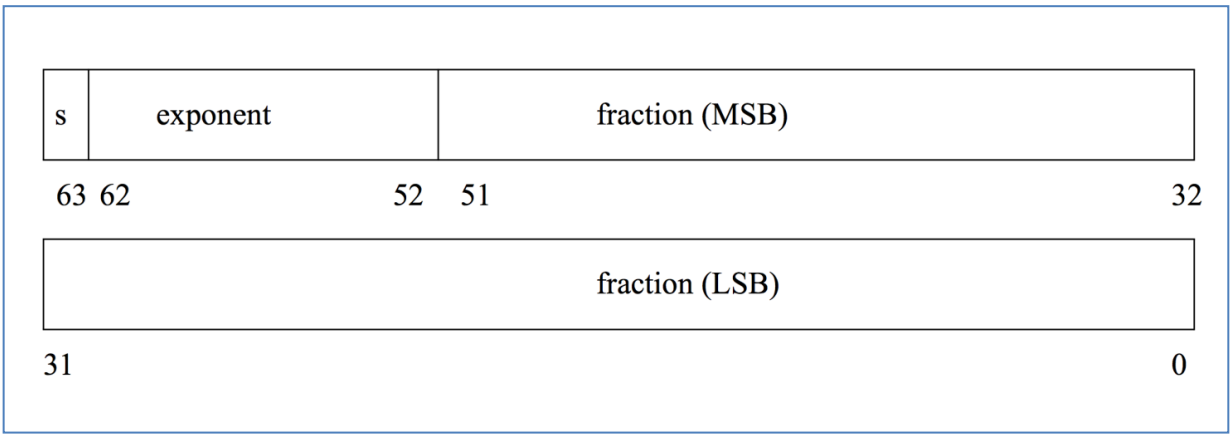
- a 52-bit fraction, 0 being the least significant bit.
- an 11-bit biased exponent
- a 1-bit sign (“s” in the figure)

The extreme absolute values which can be represented are:

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

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 6.3-3 Double-precision Floating Point Storage Format



The IEEE 754 convention for representing extreme numbers (infinity, zero, etc.) shall be used to store such values.
Date and time fields shall be represented with integers, using one of the standard transport time for- mats, as specified in [MCD] .
Angles shall be represented using the same conventions as defined in ascii-17.
Geographical locations shall be represented using the same conventions as defined in ascii-18.
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).
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

6.3.2 Standard/Commercial Binary Data Representation

The binary representation is specific to each standard /commercial binary format such as HDF5, netCDF4, BUFR, JPEF, TIFF, PNG, PDF.

7 Header 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 7.1-1 Earth Observation 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) in its file format standard tailoring document.
Notes	string			%s	Multi-lines free text. This can be used for any type of comment, relevant to that instance of the file.
Mission	string			%s	A 1-word description of the Mission, coherent with the Mission element in the File Name. Each Mission shall define the Mission description in its file format standard tailoring document.
File_Class	string			%s	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 in its file format standard tailoring document.
File_Type	string			%10s	It is a repetition of the File Type element in the File Name. Each Mission shall define the list of official file types in its file format standard tailoring document.
Validity_Period	structure				see Table 7.1-2 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).
EOFFS_Version	string			%s	Applicable version of the present document e.g. 1.4, 2.0, 3.0 … Note: Additional field does not cause any backward incompatibilities with previous version of this Standard.
Source	structure				see Table 7.1-3 for elements content

Table 7.1-2 Fixed Header - Validity Period XML Structure

Tag Name	Type	unit	Preci	C-Format	Comment

			sion		
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 Start 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 7.1-3 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, PDGS, SSALTO...)
Creator	string			%s	Name of the facility or tool, within the Ground Segment element, creating the file (e.g. MCS, DFEP, 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. 7.1-4 Earth Observation 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>TTTTTTTTTT</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>
     <EOFFS_Version>version of the applicable EOFFS Document</ EOFFS_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 (see [SCHEMA] for the correct syntax):

1. for XML data blocks, this is redundant with the reference in the XML data block itself which is mandatory
2. for non-XML data blocks (non-XML ASCII data blocks or user-defined binary data blocks), 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
- any other information needed to interpret and use the Data Block file(s) such as
  1. a list of <Data_Block_Files> records whereby the following information is provided for each Data Block files, using the tag <List_of_Data_Block_Files> according to xml-10 and xml-11.
  2. Each Data_Block_Files> records shall contain
     - The Data Block *File_Name,* including extension
     - The Data Block *Format*
     - The Data Block *Format_Ver*sion , empty tag if not relevant
     - The Data Block Schema Reference : a reference to the XML schema reference (see [SCHEMA] for the recommended syntax) allowing to validate the file
     - The Data Block *Byte_Order* , empty tag if not relevant eg for standard/commercial binary Data Block Files

Table 7.2-1 Variable Header – Data Block File XML Structure

Tag Name	Type	unit	Precision	C-Format	Comment
File_Name	string			%s	It is a repetition of the Data Block File Name including its extension.
Format	string			%s	It specifies the format of the Data Block File, In particular for standard/ commercial formats such as XML, NON XML ASCII, XLS, PDF, XML, NetCDF, HDF5, JPEG,PDF,TIFF, FITS, …
Format Version					It specifies the Format version for the format defined above, in particular for identification of evolution of standard / commercial formats An empty tag shall be used If not relevant,
Schema_Reference	string			%s	A reference to the XML schema reference (see [SCHEMA] for the recommended syntax) allowing to validate or to read the file It is relevant in particular for - user-defined binary Data Block, as the schema reference is not provided in the Data Block itself

					- non XML ASCII Data Block, as the schema reference is not provided in the Data Block itself - XML ASCII Data blocks, even if the XML schema reference is also provided as part of the Data Block itself. An empty tag shall be used if not relevant or available, eg the case of : - Standard/Commercial Binary File.
Byte_Order	string			%4s	byte ordering information, to allow compensation for byte-swapping on certain computer platforms 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 0 the standard byte order is “3210”, as defined in 0 This tag shall be used for user-defined Data Block Files and is generally not relevant for standard/commercial Data Block Files and for ASCII files .An empty tag shall be used If not relevant.

Ex. 7.2-2 Earth Observation File – Variable Header XML Syntax for Multiple Data Blocks

```
<List_of_Data_Block_Files count="number_of_elements">
     <Data_Block_File>
          <File_Name>data block #1 file name</File_Name>
          <Format>data block #1 file format</Format>
          <Format_Version>data block#1 file format version</Format_Version>
          <Schema_Reference>XML-Schema-Reference</Schema_Reference >
          <Byte_Order>3210</Byte_Order>
     </Data_Block_File >
     ...
     <Data_Block_File>
          <File_Name>data block #N file name</File_Name>
          <Format>data block #N file format</Format>
          <Format_Version>data blockN1 file format version</Format_Version>
```

```
            <Schema_Reference>XML-Schema-Reference</Schema_Reference >
            <Byte_Order>0123</Byte_Order>
        </Data_Block_File >
    </List_of_Data_Block_Files >
```

7.2.2 Variable Header Content for User-defined Binary Data Blocks

In addition to the general considerations above, in the case of a single Data Block File with user-defined binary format the fields defined in Table 7.2-1 and Ex. 7.2-2 are mandatory.

7.2.3 Variable Header Content for User-defined 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.

7.2.4 Variable Header Content for Standard/Commercial Binary Data Blocks

In addition to the general considerations above, no additional information is required in the case of a single Data Block File with standard/commercial binary format e.g. containing XLS, PDF, JPEG, PNG, TIFF, NetCDF, HDF5, FITS, etc…
However, for commonality with section 7.2.2 and Section7.2.5, the fields defined in Table 7.2-1 and Ex. 7.2-2 are recommended in this case too.

7.2.5 Variable Header Content for Multiple Data Blocks

In the case of multiple Data Block Files the fields defined in Table 7.2-1 and Ex. 7.2-2 are mandatory for each Data Block.

Ex. 7.2-3 Earth Observation File – Variable Header XML Syntax for Multiple Data Blocks

```
<List_of_Data_Block_Files count="number_of_elements">
     <Data_Block_File>
          <File_Name>data block #1 file name</File_Name>
          <Format>data block #1 file format</Format>
          <Format_Version>data block#1 file format version</Format_Version>
          <Schema_Reference>XML-Schema-Reference</Schema_Reference >
          <Byte_Order>3210</Byte_Order>
     </Data_Block_File >
     ...
     <Data_Block_File>
          <File_Name>data block #N file name</File_Name>
          <Format>data block #N file format</Format>
          <Format_Version>data blockN1 file format version</Format_Version>
          <Schema_Reference>XML-Schema-Reference</Schema_Reference >
          <Byte_Order>0123</Byte_Order>
     </Data_Block_File >
</List_of_Data_Block_Files >
```

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 Observation files. These tools are briefly introduced in the following sections, and their functionality and usage is fully described in [IO-SUM] .

9.1 File Validation

9.1.1 XML ASCII File Validation

Earth Observation 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:
    1. file structure (i.e. the sequence of tags)
    2. tag values type and allowed range
- a central repository, to be setup by each project, 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

It is recommended that schemas follow the indications given in [SCHEMA] .

9.1.2 Non-XML ASCII File Validation

Concerning non-XML parts, dedicated validation tools need to be made available, these might involve e.g. a non-XML validation schema, dedicted validation software tools.

9.1.3 User-defined Binary File Validation

Binary syntax cannot be intrinsically validated. However content of user-defined binary files can be validated e.g. by use of Schemas according to [DFDL] providing a format description and parameter range/value.

9.1.4 Standard/Commercial Binary File Validation

Standard/Commercial Binary syntax validation is performed only if the corresponding format writing/reading software/library implements it.

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:
    1. 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)
    2. file visualization as simple ASCII text, using any text tool such as a text editor; data can be pre- sented without the burden of the XML tags, in a “good old ASCII” style, which is not formal but readable
- a central repository, to be setup by each project, 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 Observation 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-processing 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)

Translations from other formats into Earth Observation 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 Observation 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 Observation files are 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 Observation file headers, transparently (i.e. without a-priori knowledge of 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 editor” for interactively reading and writing XML-based Earth Observation 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 Observation file, using WYSIWIG HMI techniques
- control of data types and allowed ranges through use of the relevant XML schemas

Software Libraries and routine to read/write binary data described using a DFDL schema are also available (not exhaustive):

- DFDL4S
- DFDL Daffodil

9.5 Header and Data Block Packaging / Un-Packaging

To support packaging and un-packaging of Header file and Data Block file(s), and optional compression and decompression, standard tool supporting the prescribed zip format are commonly available and part of standard operating systems.

The packaging using zip into a single file is also strongly recommended for data product packaged according to higher level standard for archival and long term preservation (e.g. SAFE directories) whenever these directory structures need to be transported within a ground segment