BinX Usage Standard

PE-TN-ESA-GS-120

M.Zundo (ESA/ESTEC)
A.Gutierrez (Deimos Engheneria)

1. PURPOSE AND SCOPE

Purpose of this TN is to specialise the usage standard of BinX Binary description language to :

1. describe the binary data block (DBL) of a ESA Earth Explorer data product
2. constrain the usage of BinX constructs to realise a common data interface
3. allow compatibility with existing Read/Write software libraries like the EE XML/Binary File Handling Library [BINXML_UM].

The activities which lead to the output above have been performed in the framework of the SMOS Level 1 Processor Prototype performed by Deimos and Critical software.

2. REFERENCE DOCUMENT

[BINXML_UM]	EE XML/Binary File Handling Library User Manual, SO-UM-DME-L1PP-0005, issue 1.5, 02-05-2005
[BINX_DG]	Editkt::BinX 1.2 Developer's Guide. Available at http://www.edikt.org/binx/
[EE_GS_FFS]	Earth Explorer Ground Segment File Format Standard, PE-TN-ESA-GS-0001

3. INTRODUCTION

XML Standards from W3C http://www.w3.org/ do not cater for binary data definitions, which are outside the scope of the XML.

BinX library from http://www.edikt.org/binx/ is a freely available software which mimics the behaviour of XML in order to allows description of binary data structure and format by means of an XML files which, functionally, represent a schema for the binary data but which is not a schema as per W3C definition since:

- the binary data to be described does not conform to encoding constraints for XML files.
- the describing “schema” file is not a .XSD but rather a “normal” XML one.

Obviously a normal XML parser is not able to process this kind of data and their definition and it is here that the BinX library plays a role similar to an XML parser (like Xerces for instance) allowing the user to access (read and write) data inside the binary data by mean of symbolic references contained in the XML files which make the function of a schema

The data entities we are dealing with are therefore:

- One (or multiple) binary data instance file
- One corresponding XML “schema” file (hereafter called BinX Schema)

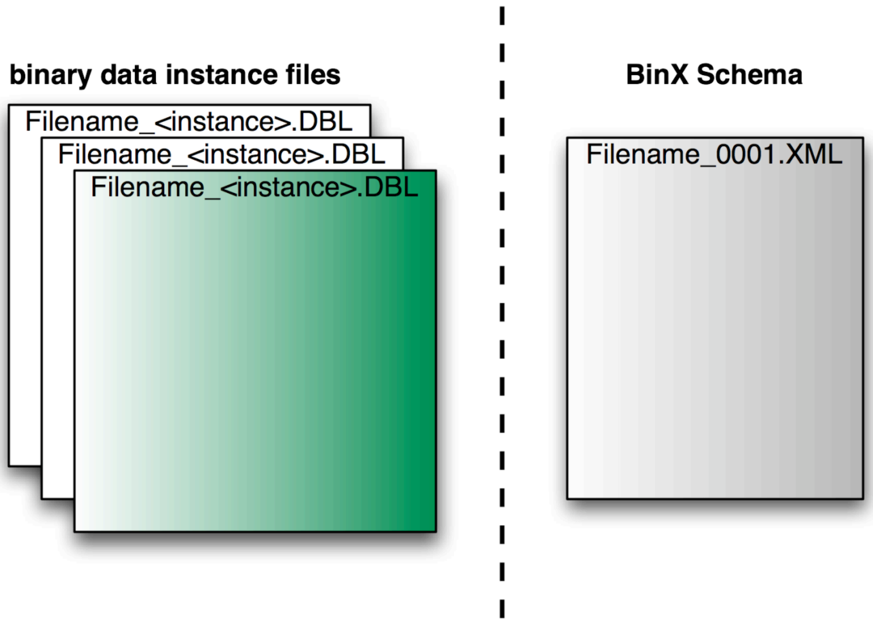

Fig 1 Binary data entities

The syntax (i.e. the name of the tags) to be used in the BinX schema are contained in [BINX_DG] and this syntax allows writing an XML file which in turn describes how the binary data file is organised and structured.

Directly using the BinX library allows to operate on the binary data only inasmuch the BinX schema contains a one to one correspondence with the actual data to be accessed. Binary files containing different repetition of same binary structure require definitions of a multiple and variable number of BinX schemas differing only in the number of element present in the data file to be used, thing this which would be absolutely unpractical.

For this reason the concept of *Binary XML Template File* has been defined (see section 4). This template definition is what we have to use when preparing a "schema" for a binary data file. (See Fig.2 below)

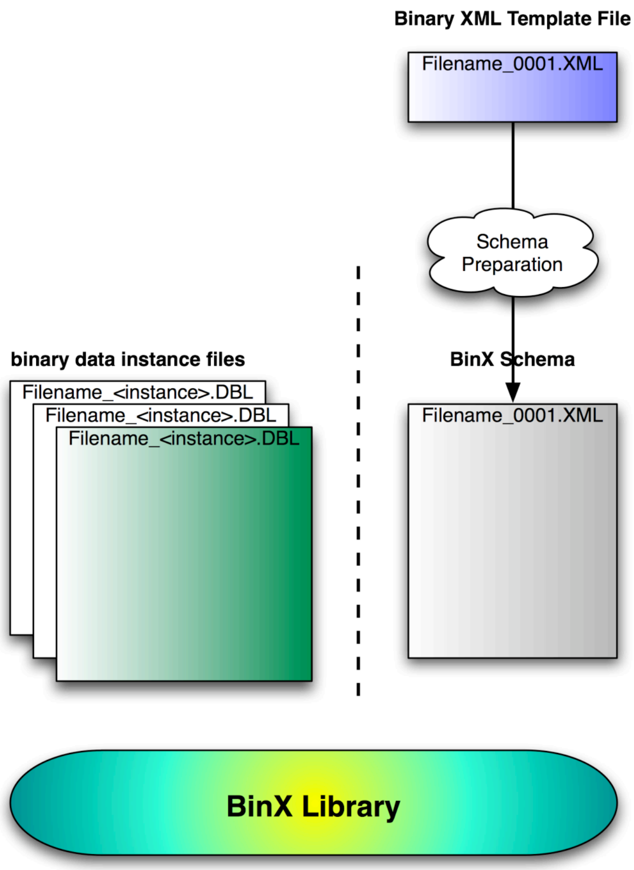

Fig. 2 Binary XML Template File expansion.

Furthermore binary data to be described is accompanied an ASCII (XML) header as described in [EE_GS_FFS] which contains metadata and which can be either joint with the binary part forming a so called hybrid file (like it has been done in SMOS L1 Processor Prototype) or kept in a separate file.

4. BINARY XML TEMPLATE FILE

The Binary XML template file is used to generate the schemas required by BinX to read the binary file.
We have to distinguish the case where use is made of the EE Binary XML library developed for SMOS and cases where this is not. It is however important to understand that the data description must be the same in both case, otherwise we would be neglecting the very benefit to use an XML schema for representation.

1. In the case use of EE Binary XML Library, this template file is used by Binary XML Library to create intermediate internal BinX schemas and to access each individual hybrid file, according to the number of binary structures that are contained.

2. In case no use is made of the Binary XML library then another ad-hoc software performing the same translation function will need to be implemented, an expansion of the template performed and the result passed to the off-the-shelf BinX library.

The template file is a XML file following the specification of BinX but with some restrictions:

- ❑ in the schema template, after the tag defined by `<Dataset>`, only the first line containing the `<useType>` tag will be taken into account when creating the schema using the EE Binary XML library or equivalent. **Any tag and/or comment after this line is ignored** so it should not be used.

- ❑ the variable name must contain the # character that identifies the place where the index of the element will be placed.

Following it is an example of a template:

```
<?xml version="1.0" encoding="UTF-8"?>
<binx xmlns="http://www.edikt.org/binx/2003/06/binx">

   <definitions>
        <defineType typeName="Basic_Types">
           <struct>
             <unsignedInteger-32 varName="var_integer" />
             <float-32 varName="var_float" />
           <double-64 varName="var_double" />
             <unsignedLong-64 varName="var_long" />
           <unsignedByte-8 varName="var_char" />
           </struct>
       </defineType>
   </definitions>

<dataset src="binary_product" byteOrder="bigEndian">
     <useType typeName="Basic_Types" varName="B_Types[#]" />
</dataset>
</binx>
```

Fig. 3 Binary XML Schema Template

5. BINARY XML EARTH EXPLORER VARIABLE HEADER REQUIREMENTS

The following section describes the required elements (mandatory tags) that must be present in the XML header of any product to allow compatibility with the existing EE Binary XML Library. These tag must be added to any product specified XML header. Due to the use of XML this operation is completely transparent to any software which does not make use of it and does not cause any change in interface or application software interpreting the data.

1. The XML header of the product must contain in the first level of indentation a specific tag with the size of the header. This tag is `<Header_Size>` and must be located within the first 10 lines of the Variable Header Part e.g. in the MPH (which is defined in [EE_GS_FFS]).

2. The XML header must also contain a list of elements repeated for each data set in the list. These elements must contain the following tags and corresponding parameter:
   - **`<Data_Set_Name>`**: Parameter describing the name of the Data_Set to be read. This must match the name given in the template.
   - **`<MDR_Size>`**: Parameter describing the individual size of each Measurement Data Record within this Data_Set (in the case of variable size arrays, the value shall not be used)
   - **`<Num_MDR>`**: Number of Measurement Data Records within the Data_Set
   - **`<MDR_Offset>`**: Offset in bytes from the beginning of the binary data block product (i.e. excluding the header) to the beginning of the binary Data_Set (identical also to the beginning of the first MDR)
   - **`<Byte_Order>`**: Endianess of the Measurement Data Records (3210 for Big Endian, 0123 for Little Endian)

In the following paragraph there is an example of such a MPH header:

```
<FILE>
   <Header_Size>0000005367</Header_Size>
   <List_of_Data_Set count="01">
      <Data_Set>
         <Data_Set_Name>CHARACTERS                  </Data_Set_Name>
         <MDR_Size>+0000002</MDR_Size>
         <Num_MDR>+00075</Num_MDR>
         <MDR_Offset>+0005367</MDR_Offset>
         <Byte_Order>3210</Byte_Order>
      </Data_Set>
   </List_of_Data_Set>
</FILE>
```

Fig. 5 Mandatory tags in the XML product Header