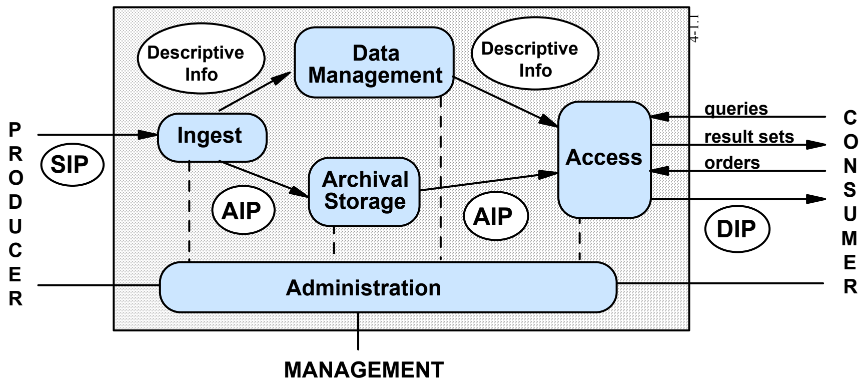
XFDU packaging contribution to an implementation of the OAIS reference model

Arnaud Lucas, Centre National d'Etudes Spatiales

18, avenue Edouard Belin
31401 Toulouse Cedex 9
FRANCE
Arnaud.lucas@cnes.fr

Abstract. The ISO Reference Model for an Open Archival Information (ISO 14721) defines various concepts for archiving digital data, in particular models of information packages to be stored in an Archive (Archival Information Package or AIP), or to be exchanged between an information Producer and the Archive (Submission information Package or SIP), or to be distributed to Archive consumers (Dissemination Information Package or DIP), as indicated in the following OAIS functional model figure.



The XFDU (XML Formatted Data Unit) is an emerging CCSDS recommendation to package data and metadata, including software into a single package to facilitate information transfer and archiving.

The purpose of this paper is to demonstrate what the XFDU packaging brings to possible implementations of the OAIS SIP, AIP and DIP models.

XFDU description

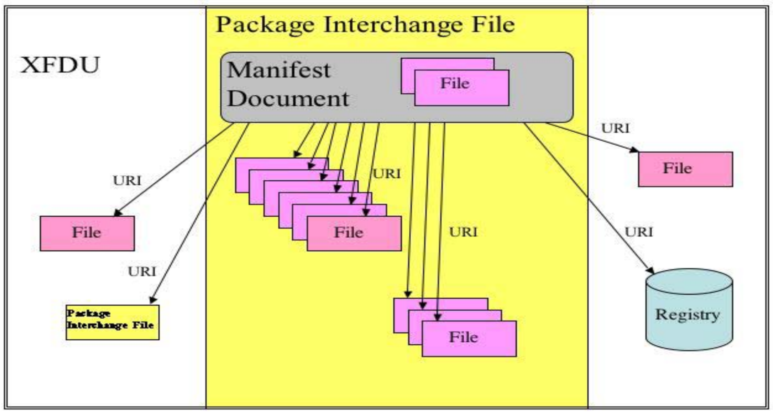


Fig. 1. XFDU logical view

An XFDU package is made of a physical container such as a ZIP file that contains a manifest file (an XML file). This manifest contains all the valuable information about the data inside the container and/or outside the container.

At the higher level, the manifest is split into in different parts:

- The Information Package Map contains the logical view of the package. It's a hierarchical xml tree representing the content of the package. Each leaf of the tree is a Content Unit and can be referred to from the other parts of the package (i.e. information is linked by the Content Unit reference ID).
- The Data Object Section contains all the physical information needed to get the file objects out.
- The Metadata Section records all of the metadata for all items in the package.
- The Behavior Section associates executable code with the content of the package.

The Content Unit provides the primary view into the package as it refers to each of the data objects and it associates appropriate metadata with each data object. The Content Unit reference to the metadata is via one or more metadata Category pointers. For each such pointer, there is a set of metadata classes that may be chosen to further classify the metadata object. The actual Metadata Object may be included in the manifest file or referenced by URI. A Content Unit may also contain other Content Units reference external XFDUs.

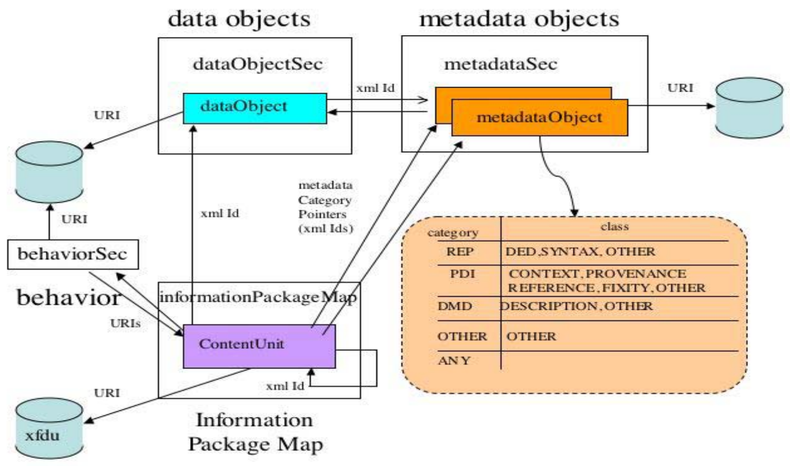


Fig. 2. Manifest file logical view

An ***XML Formatted Data Unit*** (XFDU) is the complete contents as specified by the Information Package Map (i.e., the. highest level Content Unit) component of the XML Manifest. This includes the XML Manifest document, files contained in the XML Manifest, files referenced in the XFDU Manifest including those contained within the XFDU Package, and resources (i.e., files and XFDU Packages) external to the XFDU Package. The XFDU is a logical entity and may never exist as a physical entity.

This structure allows using the XFDU packages in a lot of data transfer processes such as described in the OAIS reference model.

XFDU as SIP (Submission information Package)

First example : Linking data & metadata

In the case of packages being ingested by an archive center, XFDUs can provide obviously the link between data and metadata (embedded or available online).

If a syntactic description metadata is provided, an automatic check of the packaged data can be performed. This means that the data file will be checked against an xml schema provided by the metadata layer.

For this example, let's take an XFDU containing a XML data file (data.xml) and its associated metadata as a word document (data.doc). This data file is described by a w3c schema file located by a given URL (http://www.ccsds.org/data.xsd). When the archive receives this package, the informationPackageMap is read and the content unit "data" (the file data.xml) is given back with its associated metadata (DOC pointer) after being checked using the XSD pointer. A logical view of the XFDU package should be :

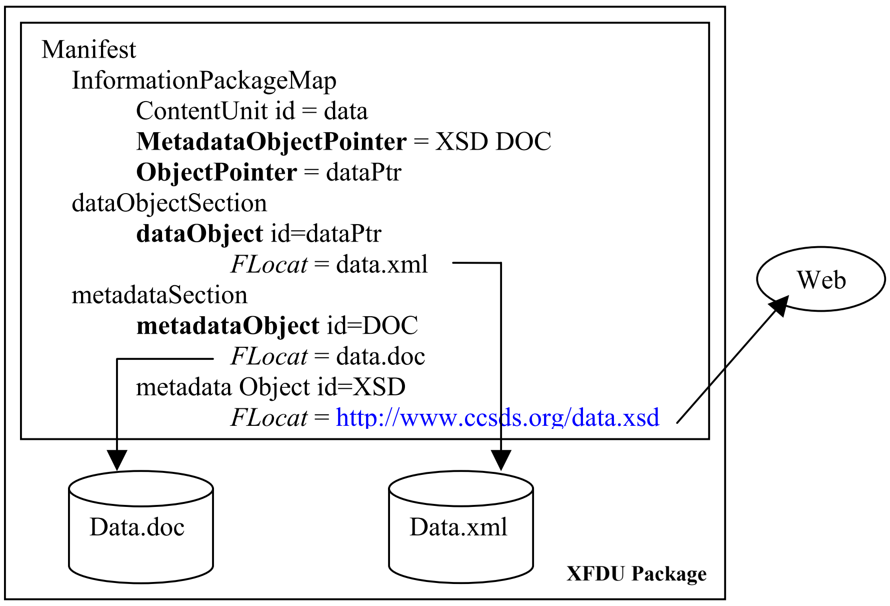
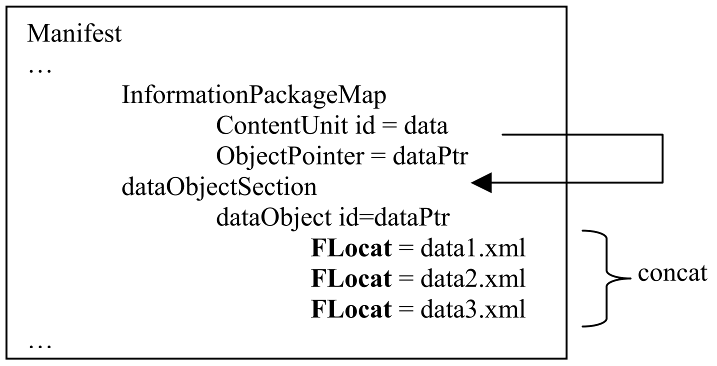


Fig. 3. Manifest file linking data & metadata

2nd example : Dealing with large files

Managing large file is not a challenge for the XFDU, because a multi-part file access is coded into the Data Object Section. In this example, a single ContentUnit named data is split into three different parts at the DataObject level (using the FLocat identifier). When receiving this package, all the parts are automatically concatenated.



```
Manifest
…
        InformationPackageMap
                ContentUnit id = data
                ObjectPointer = dataPtr
        dataObjectSection
                dataObject id=dataPtr
                        FLocat = data1.xml
                        FLocat = data2.xml         concat
                        FLocat = data3.xml
…
```

Fig. 4. Multipart files

3rd example : ordering content units

Dealing with multiple-part files highlights XFDU's order attribute, which allows conditional processing. For example, it's possible to make a logical and hierarchical view of the incoming packages, check up the integrity of the whole deposit and launch a specific action when receiving the package. In this example, the archive receives a book with an introduction and two chapters :

```
Manifest
…
        InformationPackageMap
                ContentUnit id = theBook
                        ContentUnit id = Chapter2
                                ObjectPointer = chap2Ptr
                                Order=3
                        ContentUnit id = Introduction
                                ObjectPointer = introPtr
                                Order=1
                        ContentUnit id = Chapter1
                                ObjectPointer = chap1Ptr
                                Order=2
  …
```

Fig. 5. Order and hierarchical view

4th example : Specializing information

The XFDU manifest file is a xml file driven by a W3C schema. This schema is extensible, this means that you can add your own information in the ContentUnit element. For example if you want to have a date and a comment in the manifest file, you just have to extend the XFDU schema to add these elements into a new MyContentUnit. Since MyContentUnit is derived from a ContentUnit, we have a valid manifest file. This one should look like this :

```
Manifest
…
        InformationPackageMap
                MyContentUnit id = Data
                        MyComment= "a modified Content Unit"
                        ObjectPointer = dataPtr
                        MyDate=2005/12/25
…
```

Fig. 6. Specialization of a ContentUnit

XFDU as AIP (Archival Information Package)

5th example : automatic generation of catalogues

The manifest file is a xml file. It can be processed by an XSLT stylesheet in order to produce administrative information to help managing the archive. For example, the information inside the XFDU manifest file can be processed in order to produce a catalogue in the format of the archive management system.

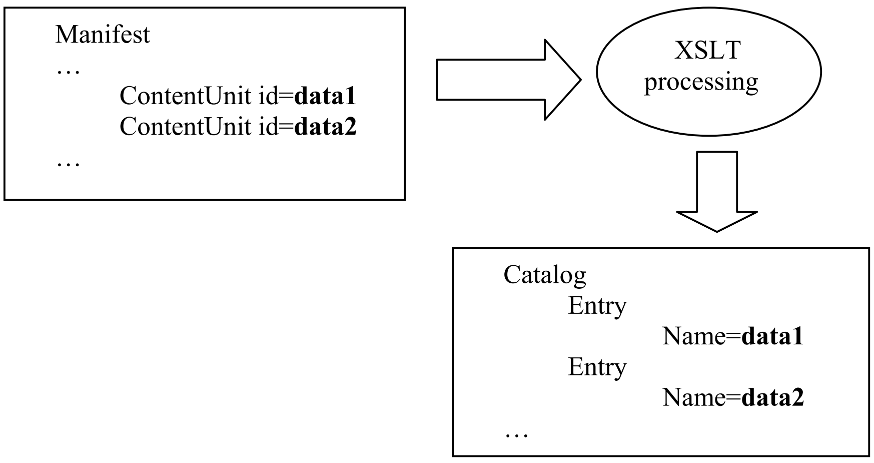


Fig. 7. Automatic transformation of the XFDU content

6th example : Optimizing an archive

Duplicated metadata can be simply grouped and accessed globally from the whole system using the XFDU external links mechanism as seen in the first example. In this example, different contentUnits in multiple XFDU packages share the same metadata.

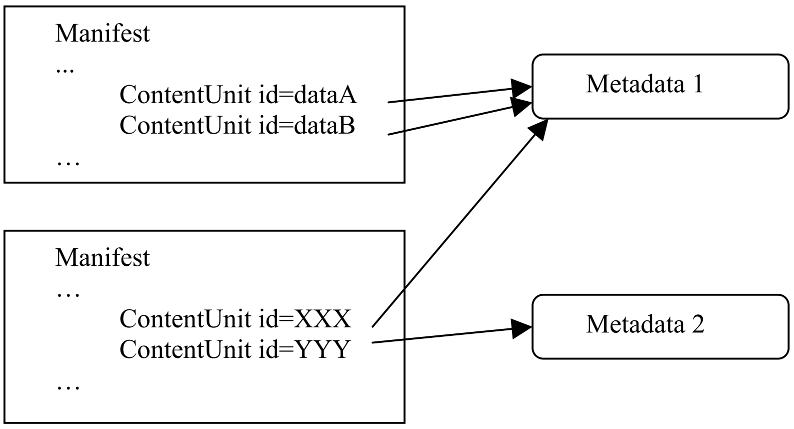


Fig. 8. Archive optimization with metadata share

7th example : Specialization for multi-mission archive

. In a multi-mission archive, XFDUs can be specialized for each mission and stay compatible at a higher level. The schema validating the manifest file can be modified as seen in example 4. This can be a great saving, using the same software to access all the data. ESA has a specialized XFDU format named SAFE. The SAFE packages have been specialized in order to manage multi-mission data (from ENVISAT instruments).

XFDU as DIP (Dissemination Information Package)

In this last part of my paper, the aim is to deliver data to the final user in a friendly way. In fact, we can take advantage of all the preceding examples and gather all the different XFDU techniques already shown.
To deliver data to the final consumer using the power of xml would be very easy (data transformation to the requested data format using XSLT). As for the SIP, dealing with large files is easy with XFDUs (automatic split can be performed). Once again, the hierarchical architecture allows the packaging of several requested products into one logical package. The metadata part brings information about the processing of the extracted data and even points to extra valuable services that could be of interest for the requester, like for example a time based extraction.

Conclusion

This paper has shown some possible use of the XFDU packages. Sure, there are still a lot to discover. The technology behind the XFDU is universal (XML and schemas) and the design of this CCSDS recommendation is made to cover a very large panel of needs, originally for the archive community but not especially.

References

CCSDS : http://www.ccsds.org
XFDU development site : http://sindbad.gsfc.nasa.gov/xfdu