

TOWARDS A CCSDS XML PACKAGING APPROACH IN THE ESA DATA DISPOSITION SYSTEM

Nestor Peccia ¹, Gianpiero Di Girolamo ¹, Michael Dyck ², Alvaro Espada ², Richard Corkill ²

¹ ESA/ESOC, Robert-Bosch-Strasse 5, D-64293 Darmstadt, Germany
Tel. +49 (6151) 90243 1, Fax +49 (6151) 903010, e-mail: Nestor.Peccia@esa.int

² Anite Systems GmbH, Robert Bosch str 7, 64293 Darmstadt, Germany

ABSTRACT

There is a great need in the future at ESA for a general purpose packaging mechanism for XML and related files. End users wish to collect, describe relationships between files, compress and package files together for easy transmission over the Web when accessing Mission Archives. They have the additional need to save the content and associated metadata such that the entire unit can be resurrected into the appropriate Databases and File Managers. XML packaging will also improve the transmission efficiency if the dynamic creation of components, within Web servers, is part of such a package.

ESA Science missions and instruments continue to produce volumes of useful data and users depend on the data systems and tools that archive this data as a means to access and analyse it. These ESA data systems (Earth Observation, Astrophysics, Space Plasma, and High Energy Physics) do not inter-operate well, are heterogeneous and geographically distributed. Users must access each data system and its corresponding science data independently through tools that have been custom-built for the particular science data system or mission. An even larger effort is to correlate these data sets across a multidisciplinary ESA effort.

New emerging technologies could provide support for a future ESA electronic data interchange system between heterogeneous data sources, i.e.:

- ❑ The Extensible Mark-up Language (XML)
- ❑ the Common Object Request Broker Architecture (CORBA)
- ❑ JAVA Programming Language

The general objective of this approach is to find out a solution to be easily and flexibly adopted by the ESA archives (in parallel to their development) and that improves their openness to other archives. The CCSDS is currently developing XML packaging standards for the space domain. ESA 's approach is based on the requirements identified by the CCSDS Panel 2, and it will provide valuable input to it by identifying the strengths and weaknesses of some of the approaches to data packaging using XM technology.

This paper demonstrates the applicability of XML technology to the packaging of data for ESA's space archive systems, and in particular by

- Analysing existing ESA archives and identifying on a selected ESA reference archive the functions that could be implemented using XML technology.
- Specifying a general purpose, flexible, powerful and highly interoperable mechanism for collecting files into a group, adding metadata about the relationships between files, compressing, encrypting, authenticating, dynamically transmitting, processing incrementally, packaging and randomly accessing XML and related files. The current packaging mechanism (CCSDS Panel 2 SFDU Standard) used by the ESOC Data Disposition System is considered in the XML approach (either by embedding or replacing it).
- Evaluating; comparing the different existing mechanism for physically packaging components (XML, ZIP, MIME, etc.)

The paper also describes the prototype developed, the mapping of its functionality with the CCSDS Open Archive Information System (OAIS) model, the emerging CCSDS standard specification on XML Packaging and the lessons learned.

Purpose

The purpose of the XML Packaging Study was to find an XML based mechanism for packaging space archive data that can be easily and flexibly adopted by existing and future ESA archives and that will improve the interoperability of archive data processing systems.

Space archive data has historically been packaged in a variety of incompatible proprietary formats ranging from the structured binary content of SFDU files to raw data files. These formats have serious limitations with respect to interoperability, ease of processing, and validation.

The XML Packaging Study has attempted to produce a packaging mechanism based on an XML vocabulary to solve these limitations. The prototypes developed in the study show the ease with which this packaging mechanism could be adopted by ESA archives.

Activities

The study included the following major activities:

Archive Survey

An analysis of existing ESA archives was carried out by ESA and provided as input to the study. The results of the archive survey, including lists of software and data formats used by archives and functions performed by archive systems, were used as the basis for analysis of XML technologies during the technology survey phase of the study. The survey data was also used to select archives, which could be used as models for the prototype development.

Consultation with CCSDS and the Space Community

During both the analysis and prototype development stages of the study, consultation with the CCSDS and members of the space community provided insight into their collective requirements and objectives with respect to XML packaging. Inputs were obtained through participation in the CCSDS XML Packaging Workshop (Washington, USA, August 2001), the XPace 2001 Workshop (Darmstadt, Germany, December 2001), and the CCSDS XML Packaging Workshop (RAL, UK, April 2002).

Technology Survey

An evaluation of XML technologies and mechanisms for packaging data was carried out. This evaluation included coverage of core XML markup as well as Namespaces in XML, DTD, XML Schema, XLink, XPath, XPointer, CSS, XSL, XSLT, XSLFO, SAX, DOM, JAXP, JAXR, JAXM, JAX-RPC, JAR, ZIP, RMI, CORBA, SOAP. The survey also evaluated several XML tools for parsing, validating, editing and processing XML.

Prototype Development

Following careful analysis of space archive system packaging requirements, the prototype development phase focused on the definition of a general purpose mechanism for packaging space archive data and associated metadata with support for services and extensibility. Effectiveness of this packaging mechanism is demonstrated with prototype applications based on the Generic GDDS model of an archive data retrieval system. The outputs of this phase included a URD, SRD, ADD, prototype software and SUM in addition to various other supporting documents all of which were collected for presentation through a single web site interface.

Technology Survey

The XML technology survey includes a comprehensive analysis of XML and related technologies. The approach to this survey was simply to identify the most common XML technologies and those

with the largest audience and greatest potential for success and to study and evaluate these technologies for their applicability to packaging space archive data.

One of the main goals of the technology survey was to identify key technologies that could be further evaluated through their employment in the prototype development phase. This goal was accomplished and at least 13 of the technologies studied in the survey work package found their way into the prototype development phase. The large number of technologies used in the prototypes illustrates the component-based nature of XML development and suggests the quantity of related yet independent technologies that must be mastered to develop system solutions using XML.

Key Technology

The technology survey drew many conclusions regarding the relevance of the XML technologies to the packaging of space archive data. From these conclusions a variety of XML technologies were chosen to be incorporated into the prototype development phase of the study to meet the challenges posed by ESA's data packaging requirements. The XML tools and technologies, which found their way into the prototypes include:

- XML Schema
- XML-Spy
- Xerces
- Xalan
- SAX
- DOM
- RMI
- XSLT
- XSLFO & FOP
- SVG & Batik
- DocBook

Key Requirements

A total of 67 software requirements were identified in the prototype SRD. Of these 67, the key packaging requirements include:

- Backward compatibility with the SFDU packaging mechanism.
- The ability to handle arbitrary binary data.
- The ability to define relationships between application data and metadata.
- Support for extensibility.
- Support for compression, authentication, encryption, and fragmentation services.
- Interoperability.

All of these key requirements are handled by the XPack Schema, which was developed for the prototypes and the prototypes demonstrate the ability and effectiveness of XPack documents to handle these requirements.

Prototype Software

The prototype software consists primarily of XML Schema files, applications written in Java and C++, XSLT stylesheets, DocBook documents and DOS command scripts. The prototypes are tied together with a web interface, which exposes all of the client side interfaces and includes all of the documentation related to the study. This is arranged in a hierarchy, which allows the user to examine related parts of the prototypes through a hierarchy of menus.

The prototypes are:

- An XML DTD and Schema for Packaging Space Archive Data (XPack).
- An API for creating and manipulating XPack documents in Java.
- Converters, which support the conversion of files between SFDU and XPack formats.

- A Query Editor, which allows XML data query requests to be created and submitted to a query retriever.
- A Query Retriever, which accepts XML query requests and obtains the requested data from an archive server.
- A Software User Manual authored in DocBook XML.

All of the Java code has been documented according to the JavaDoc standards. This has enabled an API reference to be automatically generated using the JavaDoc tool, which is part of the standard Sun JDK. The generated documentation has been integrated into the XML packaging study web site.

The XPACK Schema

The key prototype developed in the study was the XPack XML Schema. This prototype consists of a single XML file which is an XML Schema describing a vocabulary which can be used to package arbitrary data (plain text, XML, binary, etc.) and metadata and establish relationships between the data and the metadata which describes it. The vocabulary also supports services, which are commonly required in association to data packaging, including compression, encryption, authentication, fragmentation, and validation. The main element structures of the XPack Schema are shown in the following figures. The root element shown in Figure 1 is `<xpack>`, which is the main container, for a package and can be nested to allow one package to include another.

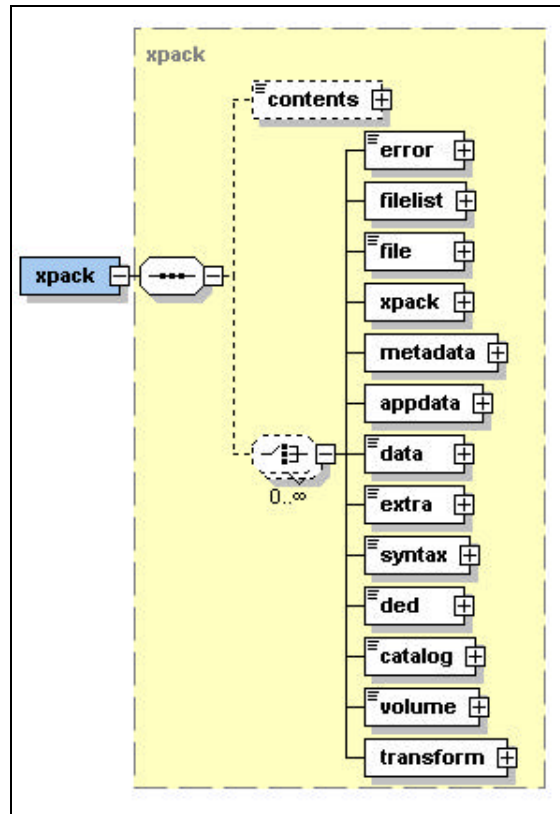


Figure 1: The `<xpack>` Element Content Model

The two most important children of the `<xpack>` element are the `<metadata>` element shown in 2, which is used for containing various types of metadata information about application data contained in the package, and the `<appdata>` element shown in Figure 3, which contains the application data itself. The relationships between `<metadata>` and `<appdata>` children are established with `mimetype`, and `subtype` attributes on these elements. Documentation of the XML Schema was accomplished with DocBook XML tags embedded directly in the schema

document. This enabled the schema documentation to be extracted by an automated process and merged with DocBook documentation for the other prototypes in preparation for publishing of the SUM. Development of an XML schema to meet a large number of requirements is a difficult task involving choices between a large number of alternatives each of which presents advantages and disadvantages to the effectiveness of the schema in meeting all of the requirements. These decisions are often compromises between aspects of the schema like flexibility vs. ease of use.

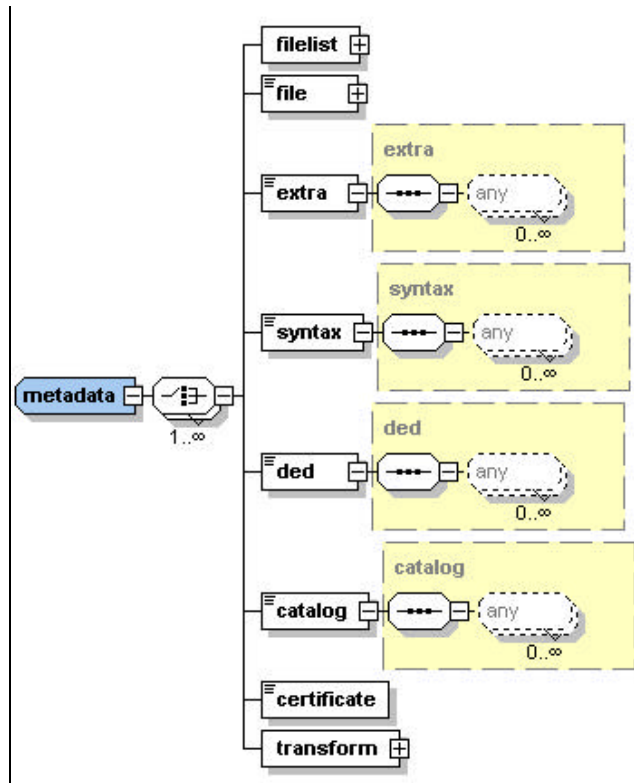


Figure 2: The <metadata> Element Content Model

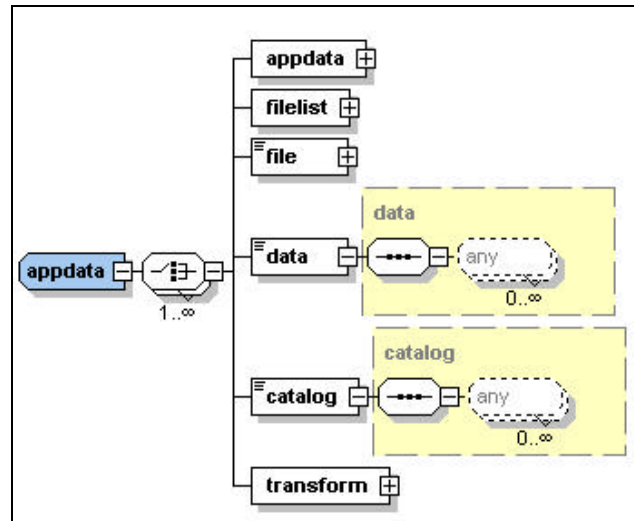


Figure 3: The <appdata> Element Content Model

XPack / SFDU Converters

The SFDU to XPack converter is written in C++ and includes support for parsing the PVL markup that is commonly found in SFDU files. Tests of the converter found that SFDU documents up to several hundred kilobytes in size could be converted into XPack documents in less than one second. Comparison of the storage requirements of converted documents depends largely on the nature of their content. SFDU tends to be more efficient for storing large quantities of binary data, whereas XPack is equally efficient when the data is predominantly text, which does not have to be encoded within the XPack document.

The XPack to SFDU converter is written in Java and provides the functionality to perform conversions in the other direction. XPack to SFDU conversion is seen as an important service to offer to customers in the event of a migration to XML based packaging since it allows users to continue to obtain SFDU results until such time as client processing software can be updated to handle XML content.

The XPack to SFDU conversion process proved equally fast, processing most of the requests tested in less than one second. The XPack to SFDU conversion can however occasionally run into problems. The current converter will only convert XPack documents that can be mapped directly to an SFDU equivalent. This means that transformations, table of contents, and other structures of XPack that are not supported by SFDU must be removed from an XPack document before a conversion is performed.

Query Editor and Query Retriever

The Query Editor and Query Retriever prototypes were developed to demonstrate the effectiveness of the XML packaging mechanism in providing the functionality typical of an archive data retrieval system like the ESOC GDDS. The prototypes clearly demonstrate the ability of the packaging mechanism to manage a heterogeneous collection of archive data types and provide an easy and flexible means for users to manipulate the data that they receive.

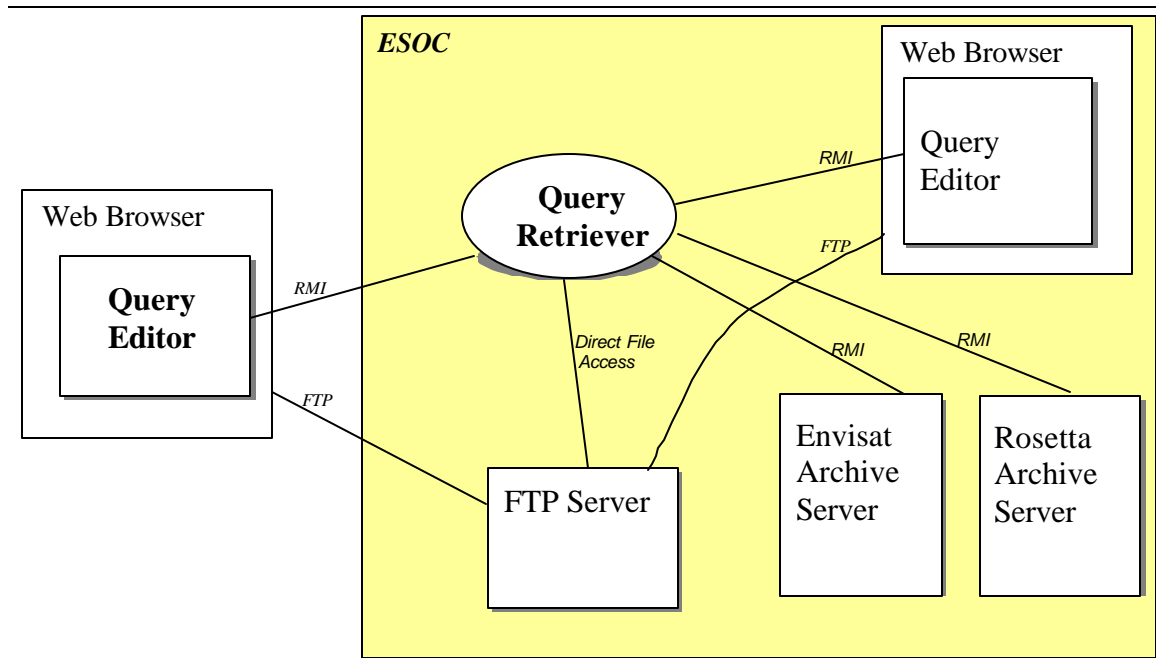


Figure 2: Relationship of XPack Query Retriever to other systems

The Query Editor allows packaging options and stylesheets to be applied in a variety of combinations. Users can retrieve archive data either in its raw format, with packaging options (such as compression) applied, with stylesheets applied, or with both packaging options and stylesheets applied. It is also possible to apply multiple stylesheets in a pipeline and to have different stylesheets specified for the results coming from different archives or from different queries from the same archive.

The stylesheet processing capabilities that XPack enables show one of the real strengths of an XML based packaging mechanism in contrast to SFDUs. Stylesheets can be used for a variety of functions including presentation, sorting and filtering. The website includes sample stylesheets for each of these functions which user's can download and try for themselves. Presentation stylesheet samples include output in several HTML styles including an alphanumeric type display. A stylesheet is also available to present results in a graphic display using SVG.

Conclusions

Effectiveness of XML for Archive Data Packaging

The XPack Schema provides a simple mechanism to enable the hierarchical packaging of data and associated metadata. In cases where the packaged data is also in an XML representation, namespaces provide an easy method for identification of interesting data within a package. The wide variety of XML processing applications available from open source initiatives makes complex processing tasks possible with third party libraries.

The Cost of XML Based Development

XML enjoys a large community of standards and open source code development, which greatly reduces the cost of deploying systems based on XML technologies.

While there are not yet a large number of tools available for editing XML documents, there are significantly more than there are for SFDU files for example and this is despite the fact that XML standard is much younger than the SFDU standard.

On the downside, XML is not very efficient with respect to memory usage and consequently the requirements for data storage, memory, and bandwidth are greater than they would be for comparable binary data representations. This is, however, a diminishing problem as hardware capabilities continue to improve while costs continue to fall.

Problems With XML Based Solutions

There were a few problems encountered during the development of the XPack prototype software. The most significant problems were found in the third part software upon which the prototypes are dependant and include the following:

- Bugs in Internet Explorer 5.5's handling of non-HTML content types. For example the display of SVG files using the Adobe plug-in works when a file is opened from windows explorer by association with IE, but fails when the same file is opened from IE directly.
- Errors in the open-source code libraries, particularly Apache Xerces support for certain DOM functions like sub tree manipulation (The ImportNode() function comes to mind). These bugs could be found and corrected in the source code, but unless the developer joins the open source forum, the fix may have to be reapplied in subsequent releases of the open source code.
- Bugs in the Sun JRE version 1.4. For example, system crashes and user interface inconsistencies in the swing file selection dialog, and numerous repaint bugs in the swing GUI controls.
- Bugs in the Microsoft Access ODBC driver (although this has nothing to do with the XML aspects of the study).

Development of effective XSL stylesheets can be very time consuming. No software was found which could provide debugging support for stylesheet development and even the smallest error in a stylesheet can result in no output, and no clues as to what went wrong. The most common technique used for debugging stylesheets therefore seems to be the practice of having the stylesheet print messages as it is being processed.

Many of the software bugs found in the very early versions of new software developed for the prototypes were related to namespace errors. It was also common to run into problems due to the combination of software, which was written to conform to different versions of various XML standards.

Recommendations for Future Development

Further analysis of the effectiveness of the XPack schema using large data volumes from existing ESA archives would be useful in clearly identifying the strengths and weaknesses of the prototype architecture.

The XPack Java API provides a strong base of code from which XPack processing applications can be developed, however a few areas for improvement and further development are noted:

- The API is currently a Java only implementation. It may be valuable to port this implementation to C++ in order to broaden the scope of platforms and applications, which can make use of the API.
- Parts of the API make inefficient use of memory. For applications, which must process huge amounts of data, the API could benefit from greater levels of stream based processing.

The XPack Query Editor and Query Retriever have demonstrated how effective the XML based packaging mechanism can be for delivering data in a distributed archive retrieval system. The transport used in the prototypes was RMI. There are some limitations with RMI however when firewalls are introduced into the network. Workarounds are available based on passing RMI messages in HTTP wrappers, but these have serious drawbacks in terms of performance and lack some of the features of RMI that are not possible due to limitations of HTTP. It would be worthwhile to examine using an alternative protocol such as CORBA, which provides security services.

The use of DocBook for documentation within ESA is highly recommended. Particularly for the Software User Manuals of web based applications. The DocBook format allows documents to be maintained from a single set of source files and then automatically converted into a wide variety of formats including plain text, HTML, PDF, Postscript, and countless others.