The Consultative Committee for Space Data Systems

Draft Recommendation for
Space Data System Standards

XML FORMATTED DATA UNIT (XFDU) STRUCTURE AND CONSTRUCTION RULES

DRAFT RECOMMENDED STANDARD

CCSDS 661.0-B-0

BLUE BOOK
January 2008

AUTHORITY

Issue:	Blue Book, Issue 0
Date:	January 2008
Location:	Not Applicable

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF AUTHORITY:)

This document has been approved for publication by the Management Council of the Consultative Committee for Space Data Systems (CCSDS) and represents the consensus technical agreement of the participating CCSDS Member Agencies. The procedure for review and authorization of CCSDS documents is detailed in the *Procedures Manual for the Consultative Committee for Space Data Systems*, and the record of Agency participation in the authorization of this document can be obtained from the CCSDS Secretariat at the address below.

This document is published and maintained by:

CCSDS Secretariat
Office of Space Communication (Code M-3)
National Aeronautics and Space Administration
Washington, DC 20546, USA

STATEMENT OF INTENT

(WHEN THIS RECOMMENDED STANDARD IS FINALIZED, IT WILL CONTAIN THE FOLLOWING STATEMENT OF INTENT:)

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of its members. The Committee meets periodically to address data systems problems that are common to all participants, and to formulate sound technical solutions to these problems. Inasmuch as participation in the CCSDS is completely voluntary, the results of Committee actions are termed **Recommended Standards** and are not considered binding on any Agency.

This **Recommended Standard** is issued by, and represents the consensus of, the CCSDS members. Endorsement of this **Recommendation** is entirely voluntary. Endorsement, however, indicates the following understandings:

- Whenever a member establishes a CCSDS-related **standard**, this **standard** will be in accord with the relevant **Recommended Standard**. Establishing such a **standard** does not preclude other provisions which a member may develop.
- Whenever a member establishes a CCSDS-related **standard**, that member will provide other CCSDS members with the following information:

  -- The **standard** itself.

  -- The anticipated date of initial operational capability.

  -- The anticipated duration of operational service.
- Specific service arrangements shall be made via memoranda of agreement. Neither this **Recommended Standard** nor any ensuing **standard** is a substitute for a memorandum of agreement.

No later than five years from its date of issuance, this **Recommended Standard** will be reviewed by the CCSDS to determine whether it should: (1) remain in effect without change; (2) be changed to reflect the impact of new technologies, new requirements, or new directions; or (3) be retired or canceled.

In those instances when a new version of a **Recommended Standard** is issued, existing CCSDS-related member standards and implementations are not negated or deemed to be non-CCSDS compatible. It is the responsibility of each member to determine when such standards or implementations are to be modified. Each member is, however, strongly encouraged to direct planning for its new standards and implementations towards the later version of the Recommended Standard.

FOREWORD

This document is a technical Recommendation to use for the packaging of data and metadata, including software, into a single package (e.g. file, document or message) to facilitate information transfer and archiving. It provides a detailed specification of core packaging structures and mechanisms that meet current CCSDS agency requirements and that augment the current CCSDS packaging and language Recommendations to accommodate the current computing environment and meet evolving requirements. This Recommendation leverages the wide community acceptance and usage of XML technologies by making the packaging manifest an XML document defined by the XML Schema specified in this Recommendation.

Through the process of normal evolution, it is expected that expansion, deletion, or modification of this document may occur. This Recommended Standard is therefore subject to CCSDS document management and change control procedures, which are defined in the *Procedures Manual for the Consultative Committee for Space Data Systems*. Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

Questions relating to the contents or status of this document should be addressed to the CCSDS Secretariat at the address indicated on page i.

At time of publication, the active Member and Observer Agencies of the CCSDS were:

Member Agencies

- Agenzia Spaziale Italiana (ASI)/Italy.
- British National Space Centre (BNSC)/United Kingdom.
- Canadian Space Agency (CSA)/Canada.
- Centre National d'Etudes Spatiales (CNES)/France.
- Deutsches Zentrum für Luft- und Raumfahrt e.V. (DLR)/Germany.
- European Space Agency (ESA)/Europe.
- Federal Space Agency (Roskosmos)/Russian Federation.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- Japan Aerospace Exploration Agency (JAXA)/Japan.
- National Aeronautics and Space Administration (NASA)/USA.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- Belgian Federal Science Policy Office (BFSPO)/Belgium.
- Central Research Institute of Machine Building (TsNIIMash)/Russian Federation.
- Centro Tecnico Aeroespacial (CTA)/Brazil.
- Chinese Academy of Space Technology (CAST)/China.
- Commonwealth Scientific and Industrial Research Organization (CSIRO)/Australia.
- Danish Space Research Institute (DSRI)/Denmark.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- Korea Aerospace Research Institute (KARI)/Korea.
- MIKOMTEK: CSIR (CSIR)/Republic of South Africa.
- Ministry of Communications (MOC)/Israel.
- National Institute of Information and Communications Technology (NICT)/Japan.
- National Oceanic & Atmospheric Administration (NOAA)/USA.
- National Space Organization (NSPO)/Taipei.
- Space and Upper Atmosphere Research Commission (SUPARCO)/Pakistan.
- Swedish Space Corporation (SSC)/Sweden.
- United States Geological Survey (USGS)/USA.

PREFACE

This document is a draft CCSDS Recommended Standard. Its 'Red Book' status indicates that the CCSDS believes the document to be technically mature and has released it for formal review by appropriate technical organizations. As such, its technical contents are not stable, and several iterations of it may occur in response to comments received during the review process.

Implementers are cautioned **not** to fabricate any final equipment in accordance with this document's technical content.

DOCUMENT CONTROL

Document	Title	Date	Status
CCSDS 661.0-R-0	XML Formatted Data Unit (XFDU) Structure and Construction Rules, Draft Recommended Standard, Issue 0	December 2006	Current draft

CONTENTS

Section Page

1 INTRODUCTION 1-1

1.1 PURPOSE AND SCOPE 1-1
1.2 RATIONALE 1-1
1.3 STRUCTURE OF THIS DOCUMENT 1-2
1.4 SECURITY CONSIDERATIONS 1-4
1.5 DEFINITIONS 1-4
1.6 CONFORMANCE 1-8
1.7 REFERENCES 1-9

2 OVERVIEW OF XFDU PACKAGING STRUCTURE 2-1

2.1 ENVIRONMENT 2-1
2.2 LOGICAL STRUCTURE 2-2

3 PHASED RELEASE DESIGN DECISIONS 3-1

4 XFDU MANIFEST COMPLEX TYPE 4-1

4.1 OVERVIEW OF XFDU MANIFEST 4-1
4.2 XML SCHEMA 4-2
4.3 UTILITY TYPES 4-3

5 PACKAGE HEADER 5-1

5.1 OVERVIEW 5-1
5.2 XML SCHEMA packageHeader Type 5-1
5.3 EXAMPLES 5-3

6 CONTENT UNIT 6-4

6.1 OVERVIEW 6-1
6.2 XML SCHEMA FOR contentUnitType 6-1
6.3 EXAMPLES 6-6
6.4 SEMANTICS 6-7

7 INFORMATION PACKAGE MAP 6-7

7.1 OVERVIEW 7-1
7.2 XML SCHEMA informationPackageMapType 7-1
7.3 EXAMPLES 7-2

7.4 SEMANTICS 7-2

CONTENTS (continued)

Section Page

8 DATA OBJECTS **8-1**
8.1 OVERVIEW 8-1
8.2 XML SCHEMA FOR DATA OBJECT TYPE 8-1
8.3 EXAMPLES 8-4
8.4 SEMANTICS 8-5
9 METADATA OBJECTS **9-1**
9.1 OVERVIEW 9-1
9.2 XML SCHEMA FOR METADATA OBJECTS 9-2
9.3 METADATA OBJECT EXAMPLES 9-4
10 BEHAVIOR SECTION AND BEHAVIOR OBJECTS **10-1**
10.1 OVERVIEW 10-1
10.2 XML SCHEMA FOR BEHAVIOR OBJECTS 10-2
10.3 EXAMPLES ..... **ERROR! BOOKMARK NOT DEFINED.**
11 FULL XML SCHEMA –NORMATIVE/RULING **11-1**

ANNEX A COMPLETE EXAMPLE XFDU **A-1**
ANNEX B UML FOR XFDU **B-1**
ANNEX C LEGEND FOR XML AUTHORITY FIGURES **C-1**
ANNEX D INFORMATIVE REFERENCES **D-1**

Deleted: ERROR! BOOKMARK NOT DEFINED.

Figure

2-1 Environment/Conceptual View of an XFDU 2-2
2-2 XFDU Manifest Logical View ..... **Error! Bookmark not defined.**
4-1 First level Decomposition of XFDUType 4-2
4-2 referenceType Schema Diagram 4-4
4-3 dataObjectPointerType Schema Diagram 4-4
4-4 fileContentType/binaryData/xmlData Schema Diagram 4-5
4-5 checksumInformationType Schema Diagram 4-6
5-1 packageHeaderType Schema Diagram 5-1
6-1 contentUnitType Schema Diagram 6-1
7-1 InformationPackageMapType Schema Diagram 7-1
8-1 dataObjectType Schema Diagram 8-1
9-1 metadataObjectType and metadataSectionType Schema Diagram 9-2
10-1 behaviorObjectType Schema Diagram 10-2
11-1 Full XFDU Schema Diagram 11-2

Deleted: Error! Bookmark not defined.

Table

3-1 XFDU Functionality by Recommendation Issue 3-1

1 INTRODUCTION

1.1 PURPOSE AND SCOPE

The main purpose of this document is to define a CCSDS Recommended Standard for the packaging of data and metadata, including software, into a single package (e.g., file or message) to facilitate information transfer and archiving. Another goal is to provide a detailed specification of core packaging structures and mechanisms that meets current CCSDS agency requirements, augment the current CCSDS packaging and language Recommended Standards (references [D1]-[D6]) to accommodate the current computing environment and meet evolving requirements, and that can be implemented to demonstrate practical, near-term results.

Deleted: [D6]

The scope of application of this document is the entire space informatics domain from operational messaging to interfacing with science archives.

1.2 RATIONALE

The current CCSDS Standards for Data Packaging have not undergone a major revision in 15 years. The computing environment and the understanding of metadata have changed radically:

– Physical media →Electronic Transfer:

  The primary form of access to, and delivery of, both archived and recently produced data products has shifted from hard media to include substantial network delivery.

– No standard language for metadata →XML:

  After 'bits' and 'ASCII', the language 'XML' can be viewed as the next universal data standard, as it has grown exponentially.

– Homogeneous Remote Procedure Call (RPC)→CORBA, SOAP:

  Communicating heterogeneous systems are increasingly using standard remote procedure calls or messaging protocols. The primary RPC and messaging protocol for the WWW is SOAP, an XML based protocol.

– Little understanding of long-term preservation→OAIS Reference Model:

  The OAIS Reference Model has become a widely adopted starting point for standardization addressing the preservation of digital information. The OAIS defines and situates within functional and conceptual frameworks the concepts of Information Packages for archiving (Archival Information Packages, or AIPs), producer submission to an archive (Submission Information Packages, or SIPs), and archives' dissemination to consumers (Dissemination Information Packages, or DIPs).

– Record formats→Self describing data formats:

  Commensurate with XML, and rapidly growing computing power and storage capabilities, there has been an increasing tendency to use data formats that are more self-describing.

Further, in the Space domain, there are a number of new requirements to facilitate such functions as being able to describe multiple encodings of a data object, and to better describe the relationships among a set of data objects. Therefore it is necessary to define a new packaging standard while maintaining the existing functionality.

1.3 STRUCTURE OF THIS DOCUMENT

This document is divided into informative and normative sections and annexes.

Sections 1-3 of this document are informative and give a high level view of the rationale, the conceptual environment, some of the important design issues and an introduction to the terminology and concepts.

– Section 1 gives purpose and scope, rationale, a view of the overall document structure, and the acronym list, glossary, and reference list for this document.

– Section 2 provides a high level view of the anticipated computing environment and the key concepts in the domain of information packaging for interchange or archiving.

– Section 3 provides an overview of the functionality of the XML Formatted Data Unit (XFDU) and it identifies additional functionality proposed for future issues of this Recommendation.

Sections 4-11 of this document are the normative portion of the specification. The primary focus is on the specification of an XML document, called the Manifest document that must also conform to the XML schema defined in this document.

– Section 4, entitled "XFDU Manifest Complex Type" is the transition from informative to normative sections. It provides a description and an XML schema diagram of the first level elements of the XFDU packaging specification material. It also discusses the "utility" types that are reused many times within the XML Schema sections of this recommendation.

– Sections 5-10 present a detailed breakdown of the important entities represented in the schema. Each section is organized in the following manner:

  - N.1 – Overview;
  - N.2 – XML schema and XML Authority diagrams;
  - N.3 – XML Example (this subsection is informative and should not be considered normative);

- N.4 –Semantics (this optional section details semantics that cannot be expressed by the W3C XML Schema Language).

There are several notation issues to be understood:

a) The W3C XML Schema fragments in sections 5-10 are not intended to be complete. The complete and ruling XML Schema for the XFDU Manifest can be found in section 11.

b) Each XML Schema section contains both an XML Authority schema diagram and the W3C XML Schema Language specification of a high level type. The developers of XML Authority Schema diagrams have decided on the following specific XML visualizations:

   – Expand declared attribute groups that are specified once and referenced many times in the W3C XML Schema Language Specification. For this reason it may appear that the XML Schema Diagrams have more specified attributes than the corresponding W3C XML Schema language specifications.

   – #wildcard is used by XML Authority schema diagrams to indicate open content and is the diagrammatic form of ##any in the XML Schema Language.

c) Since the XML Schema portions of this document only specify the XFDU Manifest the term XFDU is used rather than XFDU Manifest or xfdu Manifest. This is only true in the W3C XML Schema specification and the associated XML Authority schema diagrams.

d) To help visually clarify this Recommendation, the XML instance examples are in Times 10 point italics and XML Schema fragments are in Arial 9 point.

e) In the Definitions subsection (1.5), the use of italics indicated that the definition or acronym was initially defined in Reference Model for an Open Archival Information System (OAIS) (reference [D7]).

Deleted: [D7]

Section 11 is the full XML Schema for this specification. In the case of differences between the full Schema in section 11 and the narratives and partial schemas in prior sections, the full XML Schema is the ruling specification.

Annexes A-D are Informative:

– Annex A provides an example XFDU Manifest, parts of which are the source of the examples in sections 5-10.

– Annex B provides a Unified Modelling Language (UML) view of the XFDU Manifest.

– Annex C is a legend for symbols for the XML Authority Diagrams that appear in sections 4 through 11 of this document.

– Annex D provides Informative References

1.4 SECURITY CONSIDERATIONS

XML was chosen as the Concrete implementation language for the XFDU manifest so any or all of the XML security technologies could be applied. Also, the transformation concept allows encryption of all the files in an XFDU or encryption to be applied individually to any of the contained or referenced files.

1.5 DEFINITIONS

1.5.1 ACRONYMS AND ABBREVIATIONS

AIP	*Archival Information Package*
ASCII	American Standard Code for Information Interchange
CCSDS	Consultative Committee for Space Data Systems
CORBA	Common Object Request Broker Architecture
CRC	Cyclical Redundancy Check
DIP	*Dissemination Information Package*
ISBN	International Standard Book Number
ISO	International Organization for Standardization
METS	Metadata Encoding and Transmission Standard
MIME	Multipurpose Internet Mail Extensions
OAIS	*Open Archival Information System*
OWL	Web Ontology Language
PDI	*Preservation Description Information*
RDF	Resource Description Framework
RPC	Remote Procedure Call
SFDU	Standard Formatted Data Unit
SIP	Submission Information Package
SOAP	SOAP Service Oriented Architecture Protocol
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
W3C	World Wide Web Consortium
WWW	Worldwide Web
XFDU	XML Formatted Data Unit
XML	Extensible Markup Language

1.5.2 TERMINOLOGY

Archival Information Package (AIP): An Information Package, consisting of the Content Information and the associated Preservation Description Information (PDI), which is preserved within an OAIS.

Behavior Object: Contains an interface definition element that represents an abstract definition of the set of behaviors and (in future versions of this recommendation) one or more mechanisms that is a module of executable code that implements and runs those interfaces.

Behavior Section: A container of zero or more behavior objects.

Collection: Refers to Components that are gathered together. This is analogous to files on a file system.

Component: An object (i.e., file) that can be grouped together to be part of a Collection, or Package.

Content Data Object: The Data Object, which together with associated Representation Information is the original target of preservation.

Content Information: The set of information that is the original target of preservation. It is an Information Object comprised of its Content Data Object and its Representation Information. An example of Content Information could be a single table of numbers representing, and understandable as, temperatures, but excluding the documentation that would explain its history and origin, how it relates to other observations, etc.

Context Information: The information that documents the relationships of the Content Information to its environment. This includes why the Content Information was created and how it relates to other Content Information objects.

Content Objects: The data and/or metadata objects, and any Content Units, logically within a given Content Unit.

Content Unit: XML Structure that contains pointers to data objects and associated metadata objects, and possibly other Content Units.

Data: A reinterpretable representation of information in a formalized manner suitable for communication, interpretation, or processing. Examples of data include a sequence of bits, a table of numbers, the characters on a page, the recording of sounds made by a person speaking, or a moon rock specimen.

Data Dictionary: A formal repository of terms used to describe data.

Data Object: Contains some file content and any data required to allow the information consumer to reverse any transformations that have been performed on the object and restore it to the byte stream intended for the original designated community and described by the Representation metadata pointed to by the repID attribute of the data object.

Data Object Section: Contains a number of dataObject elements.

Description Data Unit: A Content Unit where all the content objects are metadata objects.

Descriptive Information: The set of information, consisting primarily of Package Descriptions, which is provided to Data Management to support the finding, ordering, and retrieving of OAIS information holdings by Consumers.

Designated Community: An identified group of potential Consumers who should be able to understand a particular set of information. The Designated Community may be composed of multiple user communities.

Digital Object: An object composed of a set of bit sequences.

Dissemination Information Package (DIP): The Information Package, derived from one or more AIPs, received by the Consumer in response to a request to the OAIS.

Finding Aid: A type of Access Aid that allows a user to search for and identify Archival Information Packages of interest.

Fixity Information: The information that documents the authentication mechanisms and provides authentication keys to ensure that the Content Information object has not been altered in an undocumented manner. An example is a Cyclical Redundancy Check (CRC) code for a file.

Information: Any type of knowledge that can be exchanged. In an exchange, it is represented by data. An example is a string of bits (the data) accompanied by a description of how to interpret a string of bits as numbers representing temperature observations measured in degrees Celsius (the representation information).

Information Object: A Data Object together with its Representation Information.

Information Package: The Content Information and associated Preservation Description Information that is needed to aid in the preservation of the Content Information. The Information Package has associated Packaging Information used to delimit and identify the Content Information and Preservation Description Information.

Information Package Map: Outlines a hierarchical structure, for the original object being encoded, by using a series of nested contentUnit elements. The information Package map is equivalent to highest level Content Unit included in the XFDU.

Manifest: A document containing metadata about Components, and the relationships among them. This information is stored as a Component, using an XML language designed for just this purpose.

Metadata: Data about other data.

Metadata Section: Contains or References all of the static metadata for all items in the XFDU package.

Open Archival Information System (OAIS): An archive, consisting of an organization of people and systems, that has accepted the responsibility to preserve information and make it available for a Designated Community. It meets a set of responsibilities, as defined in subsection 3.1 of the OAIS Reference Model that allows an OAIS archive to be distinguished from other uses of the term 'archive'. The term 'Open' in OAIS is used to imply that this Recommendation and future related Recommendations and standards are developed in open forums, and it does not imply that access to the archive is unrestricted.

Package: A collection that is bundled together, or packaged, into one file using a defined packaging scheme. All Packages are Collections, but not all Collections have been packaged, so they are not all Packages.

Package Header: contains metadata that apply to the whole XFDU Package. This metadata may include data to inform the XFDU parsing software about volume metadata (e.g., logical volume information and specification version), administrative metadata (e.g., author and creation data) and technical data (e.g., hardware and operating system).

Package Interchange File: A collection of files that have been bundled together into a single container that also contains a manifest describing the contained files and the relationships among those files.

Physical Object: An object (such as a moon rock, bio-specimen, microscope slide) with physically observable properties that represent information that is considered suitable for being adequately documented for preservation, distribution, and independent usage.

Preservation Description Information (PDI): The information which is necessary for adequate preservation of the Content Information and which can be categorized as Provenance, Reference, Fixity, and Context information.

Process Description Unit: Contains a description that can range from an automated scripting language to an English language description of the steps a person /intelligent agent would take in performing a process.

Provenance Information: The information that documents the history of the Content Information. This information tells the origin or source of the Content Information, any changes that may have taken place since it was originated, and who has had custody of it since it was originated. Examples of Provenance Information are the principal investigator who recorded the data, and the information concerning its storage, handling, and migration.

Reference Information: The information that identifies, and if necessary describes, one or more mechanisms used to provide assigned identifiers for the Content Information. It also provides identifiers that allow outside systems to refer, unambiguously, to a particular Content Information. An example of Reference Information is an ISBN.

Representation Information: The information that maps a Data Object into more meaningful concepts. An example is the ASCII definition that describes how a sequence of bits (i.e., a Data Object) is mapped into a symbol.

Representation Network: The set of Representation Information that fully describes the meaning of a Data Object. Representation Information in digital forms needs additional Representation Information so its digital forms can be understood over the Long Term.

Structure Information: The information that imparts meaning about how other information is organized. For example, it maps bit streams to common computer types such as characters, numbers, and pixels and aggregations of those types such as character strings and arrays.

Submission Information Package (SIP): An Information Package that is delivered by the Producer to the OAIS for use in the construction of one or more AIPs.

Transformation: A modification to the encoding of a file that is done independent of the definition of the Representation Information. A transformation is usually done to satisfy end-to-end system requirements such as performance or security. A transformation must be reversed to restore the original digital object.

Transformation Object: The part of a Data Object that contains required information (e.g., algorithms and parameters) to reverse any transformations applied to the digital content and restore that to the original binary data object.

XFDU Manifest: A Manifest that is conformant to the XML Schema specified in this Recommendation.

XFDU Package: A Package Interchange File that contains an XFDU Manifest and is conformant to the semantics specified in this document. An XFDU Package is a specialization of Package Interchange File.

XML Formatted Data Unit (XFDU): 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.

XML Schema: W3C schema specification for XML documents using XML syntax.

1.6 CONFORMANCE

An ***XFDU Manifest*** is a Manifest that is conformant to the XML Schema definition specified in this Recommendation.

An ***XFDU Package*** is a Package Interchange File that contains an XFDU Manifest linking all files of the package.

1.7 REFERENCES

The following documents contain provisions which, through reference in this text, constitute provisions of this Recommended Standard. At the time of publication, the editions indicated were valid. All documents are subject to revision, and users of this Recommended Standard are encouraged to investigate the possibility of applying the most recent editions of the documents indicated below. The CCSDS Secretariat maintains a register of currently valid CCSDS Recommended Standards.

[1] Tim Bray, et al., eds. Extensible Markup Language (XML) 1.0. 2nd ed. W3C Recommendation. N.p.: W3C, October 2000. <http://www.w3.org/TR/2000/REC-xml-20001006>

[2] Henry S. Thompson, et al., eds. XML Schema Part 1: Structures. W3C Recommendation. N.p.: W3C, May 2001. <http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/>

[3] Paul V. Biron and Ashok Malhotra, eds. XML Schema Part 2: Datatypes. W3C Recommendation. N.p.: W3C, May 2001. <http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/>

[4] [XML Authority] "TIBCO Turbo XML™." *TIBCO.com*. <http://www.extensibility.com/software/metadata/turboxml.jsp> (12/18/2006)

NOTE – Informative references are listed in annex D.

2 OVERVIEW OF XFDU PACKAGING STRUCTURE

This section provides an overview of some of the key concepts that are incorporated in the design of the XFDU packaging recommendation.

2.1 ENVIRONMENT

Figure 2-1 illustrates an abstract package in a generic computing environment to provide a basis for discussion of concepts relevant to this document. The focus of this diagram is a collection of physical files that have been bundled together because of some interrelationship. Several files have been bundled into a single container called a Package Interchange File with a specially named file called a Manifest Document included as one of the contained files. The Manifest Document describes the relations among the files and indexes the locations of all the files within the Package Interface File containing data and metadata. The Manifest can also contain data and metadata files.

The Manifest Document also references external files. The external files are shown as coming from various resources including other XFDUs, file systems and registries/repositories. In an environment with sufficient connectivity, reliability, and bandwidth the exchange of Package Interchange Files that include pointers to resources outside of the package allows the recipient to deal with the externally referenced files at its convenience. The resolution of these pointers is beyond the scope of this recommendation.

The term ‘XML Formatted Data Unit’ or ‘XFDU’ is to be understood as referring to not only the Package Interchange File and those files contained within, but also to all the external files and packages referenced from within the included Manifest file. The entire figure represents a single XFDU instance.

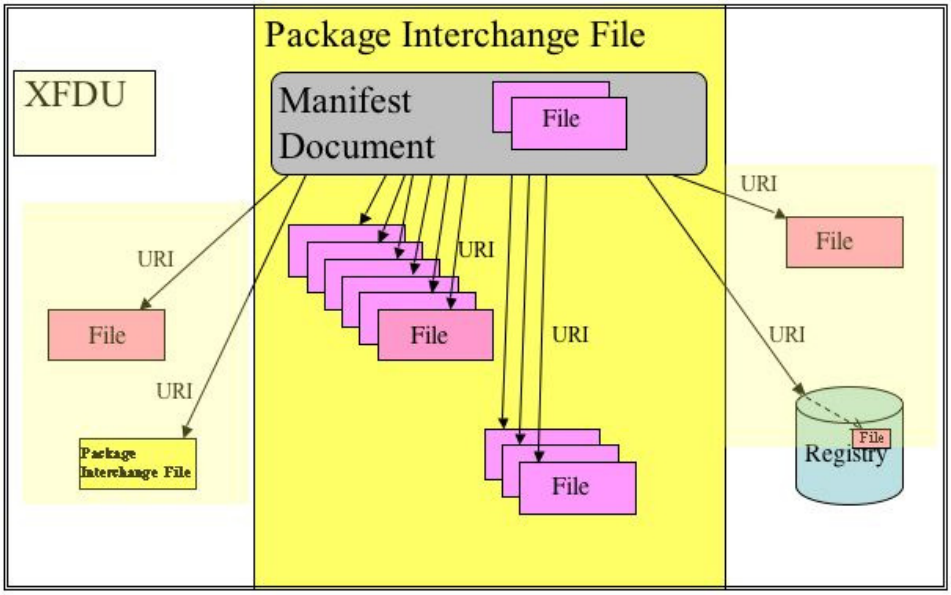


Figure 2-1: Environment/Conceptual View of an XFDU

2.2 LOGICAL STRUCTURE

This section maps the Conceptual View presented in the previous section to the terms and concepts used within the normative sections of this Recommendation.

Three high level entities are discussed in figure 2-1: the Manifest, the Package Interchange File and the XFDU. The normative sections of this Recommendation specify a concrete implementation of these conceptual high level entities. These entities are mapped into implementation specific entities as follows:

An ***XFDU Manifest*** is a Manifest that is conformant to the XML Schema specified in this Recommendation (see section 11).

An ***XFDU Package*** is a Package Interchange File that contains an XFDU Manifest and is conformant to the semantics specified in this document. figure **Error! Reference source not found.** provides an expanded view of the XFDU Manifest document showing the key entities and the possible references among them. The arrowheads show the direction of the references (e.g., the contentUnit entity references the dataObject entity). 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

Deleted: Error! Reference source not found.

may also contain other Content Units referencing external XFDUs. The figure also introduces the names and XML Labels for some of the XML entities that are discussed in the next section.

A simple structure which groups together all the information about each Information Object would work for a few objects but would lead to implementation difficulties when one has large numbers of large objects. A number of techniques are used to help alleviate the potential problems and to simplify further extraction, processing and repackaging of information contained in a package. Similar types of components are grouped into Sections such as the metadataSection in order to help simplify parsing and referencing implementations. The wrapping of the referencing pointers allows uniform access by URI to information whether it is located within the package or outside. The XFDU Manifest allows the structure of the package to be viewed without having to parse the full structure.

An ***XML Formatted Data Unit*** (XFDU) consists of the XFDU Manifest, all files contained in the Manifest and all files and XFDUs referenced from it. Some or all of the referenced files may be contained in an XFDU Package, such as through the use of a ZIP file. However there may still be references in the Manifest to files outside the XFDU Package. In this case, the XFDU is a logical entity and does not exist as a single physical entity.

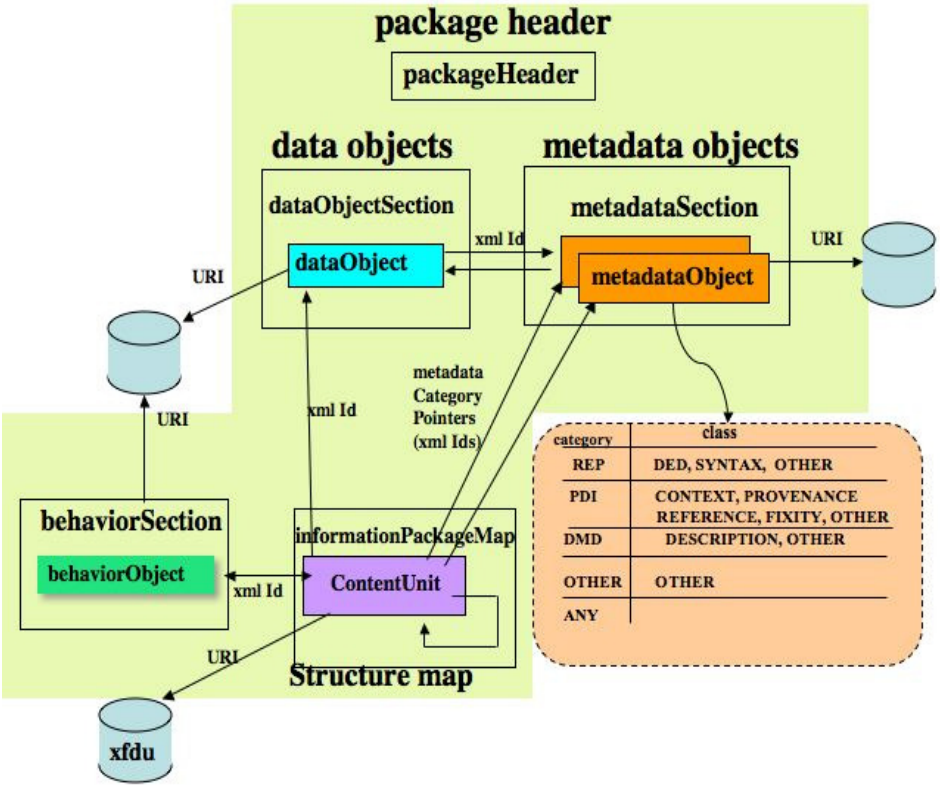


Figure 2-2: XFDU Manifest Logical View

Deleted: 3

3 PHASED RELEASE DESIGN DECISIONS

The following table provides a brief summary the functionality that should be supported by the XFDU packaging recommendation and the allocation of these requirements to this and future issues of this Recommendation. If there is no entry in the Future Issues column all anticipated functionality is included in Issue 1.0.

Table 3-1: XFDU Functionality by Recommendation Issue

Function/Feature	Version 1.0	Future Versions
Packaging techniques	1. Single XML Document 2. Archive File (e.g., tar, zip)	XML messaging form (e.g., SOAP with Attachments, XOP)
Manifest	Mandatory	
Format Description Types	1. Markup Languages (XML and vocabularies) 2. MIME types, 3. Self describing formats 4. Detached data descriptions (e.g., EAST)	
Metadata/data linkage options	1. Inclusion in Manifest as base64 or XML, 2. Referenced directly as binary or XML 3. Referenced or included as Data Object	
Relationship Description	1. Unit types indicate predefined relationships 2. Classification of metadata pointers 3. User defined metadata model support 4. Predefined support of OAIS Information model 5. Xlink attributes	Formal Description Language • RDF • OWL
Behaviors	1. Description of Abstract Interfaces 2. Abstract element for mechanisms and substitution group to enable compatibility in future versions	1. Inclusion of, or reference to, specific mechanisms/methods 2. Invoking behavior as value of content units 3. Scripting Behaviors
Extensibility	1. Element substitution using XML Schema substitutionGroup 2. Use of XML wildcards with namespace = other for extension	1. Type Substitution using xsi: type
Encodings and Transformations	The ability to allow/reverse multiple transformations on files	
Instance Validation	1. XML schema type validation 2. Enumerated lists 3. Constraints and business rules using Schematron	Evolving enumerated lists

4 XFDU MANIFEST COMPLEX TYPE

4.1 OVERVIEW OF XFDU MANIFEST

The following are brief descriptions of the five complex types/elements that may be contained in an ***XFDU Manifest*** (XFDUType). A high level XML Authority (reference [4]) XML schema diagram of the XFDU is shown as figure 4-1.

a) **Package Header** (packageHeader element of packageHeaderType): contains metadata that apply to the whole XFDU Package. This metadata may include data to inform the XFDU parsing software about volume metadata (e.g., logical volume information and specification version), administrative metadata (e.g., author and creation data) and technical data (e.g.., hardware and operating system).

b) **Information Package Map** (informationPackageMap element of informationPackageMapType): provides a hierarchical view of the content of the XFDU using a series of nested **contentUnit** elements. Content units contain pointers to data objects and to the metadata associated with those data objects.

c) **Data Object Section** (dataObjectSection element of dataObjectSectionType): contains any number of dataObject elements. A Data Object logically contains a byte stream and any data required to allow the information consumer to reverse any transformations. Reversing the transforms will restore the byte stream to the original format described by the Representation metadata pointed to from the Content Unit.

d) **Metadata Section** (metadataSection element of metadataSectionType): records all of the metadata for all items in the XFDU package. Multiple metadata objects are allowed so that the metadata can be recorded for each separate item within the XFDU object. The metadata schema allows the package designer to define any metadata model by providing attributes for both metadata categories and a classification scheme for finer definition within categories. The XFDU also provides predefined metadata categories and classes via enumerated attributes that follow the OAIS information model.

e) **Behavior Section** (behaviorSection element of behaviorSectionType): may contain any number of behavior objects. Each behavior object can be used to associate executable behaviors with one or more Content Units in the containing XFDU. A behavior object contains an interface definition element that represents an abstract definition of the set of behaviors and (in future issues of this Recommendation) a mechanism that either contains or references a module of executable code that implements and runs those interfaces.

4.2 XML SCHEMA

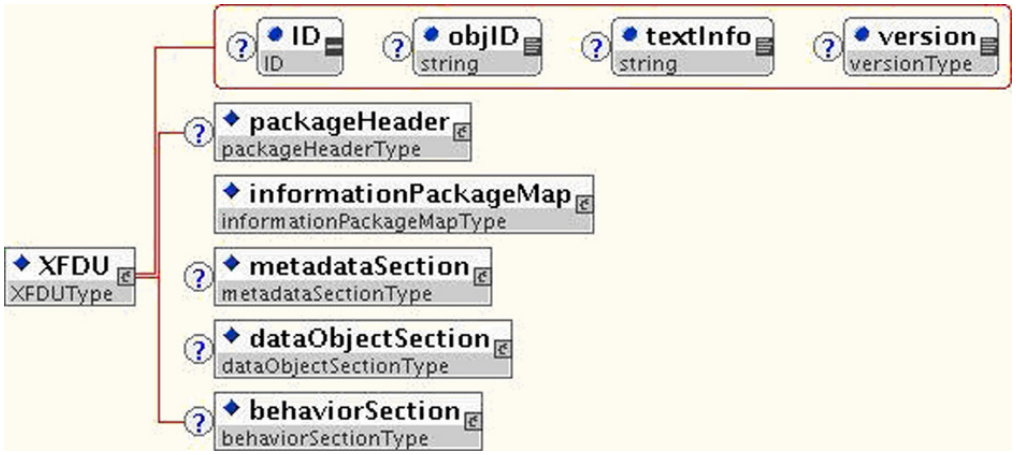


Figure 4-1: First level Decomposition of XFDUType

```
<xsd:complexType name="XFDUType">
  <xsd:annotation>
    <xsd:documentation>
    XFDUType Complex Type.
    A XFDU document consists of five possible subsidiary sections:
    packageHeader (XFDU document header), informationPackageMap (content unit section),
    metadataSection (container for metadata objects),
    dataObjectSection (data object section),behaviorSection (behavior section).
    It also has possible attributes:
    1. ID (an XML ID);
    2. objID: a primary identifier assigned to this XFDU instance by the producer of the XFDU
    3. textInfo: a title/text string identifying the document for users;
    4. version: version of the XFDU XML Schema this XFDU should be validated against. Currently this is a string but
      when formal CCSDS XML Schema Naming and Versioning rules are defined it is expected that this type will be
      specialized to conform to those rules
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="packageHeader" type="xfdu:packageHeaderType" minOccurs="0"/>
    <xsd:element name="informationPackageMap" type="xfdu:informationPackageMapType"/>
    <xsd:element name="metadataSection" type="xfdu:metadataSectionType" minOccurs="0"/>
    <xsd:element name="dataObjectSection" type="xfdu:dataObjectSectionType" minOccurs="0"/>
    <xsd:element name="behaviorSection" type="xfdu:behaviorSectionType" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="ID" type="xsd:ID"/>
  <xsd:attribute name="objID" type="xsd:string"/>
  <xsd:attribute name="textInfo" type="xsd:string"/>
  <xsd:attribute name="version" type="xfdu:versionType"/>
</xsd:complexType>
<xsd:element name="XFDU" type="xfdu:XFDUType"/>
```

4.3 UTILITY TYPES

4.3.1 OVERVIEW

These complex types are reused throughout the XFDU schema to represent some recurring structures:

- extensionType – An complex type that allows third parties to define extensions to the XFDU from a namespace that the third party owns. This type insulates the extension elements from the non-deterministic properties of the “any wildcard”.
- referenceType - Entity that can reference a resource via a URI;
- dataObjectPointerType- An empty element that references dataObjects using the XMLID;
- fileContentType- Complex type that encapsulates either binary or XML arbitrary content;
  - binaryData - Complex type that encapsulates base64 encoded data (e.g., binary data);
  - xmlData - Complex type that encapsulates 1 to many pieces of arbitrary XML data;
- checksumInformationType - Complex type that identifies the checksum type.

These simple types are reused throughout the XFDU schema to represent some recurring structures:

- mimeTypeType - attribute containing the MIME type;
- xmlDataType - a wrapper to include arbitrary XML data;
- locatorTypeType - specifies a type of location ('URL' or 'OTHER').

These attribute groups are used throughout the XFDU schema:

- LOCATION - attributes dealing with location of referenced objects;
- RegistrationGroup - attributes allowing identification of a registered item;
- xlink:simpleLink - the normal simple xlink attributes (not detailed below).

As discussed in 1.3, the XML Authority schema diagrams below show details of the XML attribute groups not shown in the associated W3C XML Schema language segments in the text. The full schema in section 11 provides the specification of these attribute groups.

4.3.2 XML SCHEMAS

extensionType

Figure 4-2:extensionType Schema Diagram

Deleted: 3

```
<xsd:complexType name="extensionType">
  <xsd:annotation>
    <xsd:documentation>
      allows third parties to define extensions to the XFDU from a namespace
      controlled by the third  party
    </xsd:documentation>
  </xsd:annotation
  <xsd:sequence>
    <xsd:any namespace="##other" processContents="lax"/>
  </xsd:sequence>
  <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>
```

referenceType

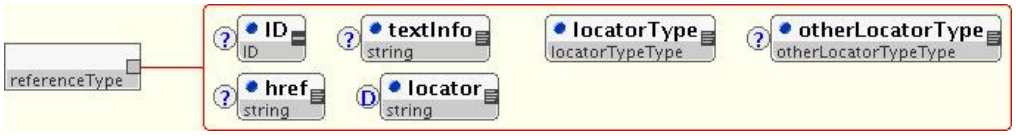


Figure 4-3: referenceType Schema Diagram

Deleted: 4

```
<xsd:complexType name="referenceType">
    <xsd:annotation>
      <xsd:documentation>
        locator attribute allows finer granularity within location specified in href
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="ID" type="xsd:ID"/>
    <xsd:attribute name="textInfo" type="xsd:string"/>
    <xsd:attributeGroup ref="xfdu:LOCATION"/>
    <xsd:attribute name="href" type="xsd:string"/>
    <xsd:attribute name="locator" type="xsd:string" use="optional" default="/"/>
  </xsd:complexType>
```

Figure 4-4: dataObjectPointerType Schema Diagram

Deleted: 5

```
<xsd:complexType name="dataObjectPointerType">
  <xsd:annotation>
    <xsd:documentation>
      The dataObjectPointerType is a type that can be used to reference dataObjects by dataObjectID.
      The dataObjectPointerType has two attributes:
```

```
        1. ID: an XML ID for this element; and
        2. dataObjectID: an IDREF to a dataObject element
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="ID" type="xsd:ID"/>
    <xsd:attribute name="dataObjectID" use="required" type="xsd:IDREF"/>
  </xsd:complexType>
```

fileContentType

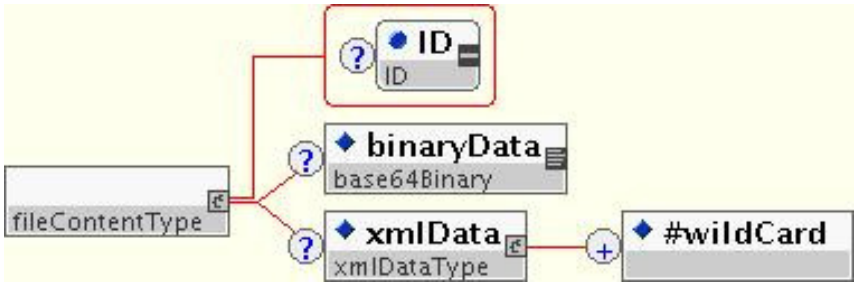
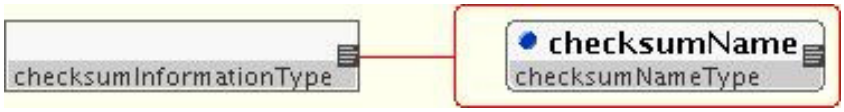


Figure 4-5: fileContentType/binaryData/xmlData[1] Schema Diagram

Deleted: 6

```
<xsd:complexType  name = "fileContentType">
    <xsd:annotation>
      <xsd:documentation>
        fileContentType encapsulates and aggregates a type that can have a choice of either
        binary or xml data
      </xsd:documentation>
    </xsd:annotation>
    <xsd:choice>
      <xsd:element name = "binaryData" type = "xsd:base64Binary" minOccurs = "0">
        <xsd:annotation>
          <xsd:documentation>A wrapper to contain Base64 encoded metadata.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name = "xmlData" type = "xfdu:xmlDataType" minOccurs = "0"/>
    </xsd:choice>
    <xsd:attribute name = "ID" type = "xsd:ID"/>
</xsd:complexType>fileContent
```

checksumInformationType



[1] #wildcard is used by XML Authority schema diagrams to indicate open content and is the diagrammatic form of ##any in XML Schema l

Figure 4-6: checksumInformationType Schema Diagram

Deleted: 7

```
<xsd:complexType name="checksumInformationType">
    <xsd:annotation>
      <xsd:documentation>
        An element of this type would convey checksum information:
        The value of the checksum element is the result of the checksum
        The value of the checksumName attribute is the name of checksum algorithm used to compute the value
      </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="checksumName" type="xfdu:checksumNameType" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
  <xsd:simpleType name="checksumNameType">
    <xsd:restriction base="xsd:string">
    </xsd:restriction>
  </xsd:simpleType>
```

The following XML classes are also used throughout this document but cannot be illustrated by an XML Authority Schema diagram because they are XSD simple types or XSD attribute groups.

mimeTypeType

```
<xsd:simpleType name="mimeTypeType">
    <xsd:restriction base="xsd:string">
    </xsd:restriction>
  </xsd:simpleType>
```

xmlDataType

```
<xsd:complexType name = "xmlDataType">
    <xsd:annotation>
      <xsd:documentation>A wrapper to contain arbitrary XML content.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:any namespace="##any" processContents = "lax" maxOccurs = "unbounded"/>
    </xsd:sequence>
</xsd:complexType>
```

LOCATION

```
<xsd:attributeGroup name="LOCATION">
    <xsd:annotation>
      <xsd:documentation>
        This attribute group aggregates attributes that can be used for specifying type of location
        This group includes following attributes:
        locatorType specifies location type (URL or OTHER)
        otherLocatorType specifies location type in case locatorType has value of OTHER
      </xsd:documentation>
    </xsd:annotation>
```

```
    <xsd:attribute name="locatorType" use="required" type="xfdu:locatorTypeType">
    </xsd:attribute>
    <xsd:attribute name="otherLocatorType" type="xfdu:otherLocatorTypeType"/>
</xsd:attributeGroup>
```

locatorTypeType

```
<xsd:simpleType name="locatorTypeType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="URL"/>
      <xsd:enumeration value="OTHER"/>
    </xsd:restriction>
</xsd:simpleType>
```

otherLocatorTypeType

```
<xsd:simpleType name="otherLocatorTypeType">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
```

registrationGroup

```
<xsd:attributeGroup name="registrationGroup">
    <xsd:annotation>
      <xsd:documentation>
        This attribute group aggregates attributes that can be used for specifying
        registration information.
        This group includes following attributes:
        registrationAuthority - the authority that issued the registration
        registeredId - the id for the registration
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="registrationAuthority" type="xsd:string"  use="optional"/>
    <xsd:attribute name="registeredID" type="xsd:string" use="optional"/>
</xsd:attributeGroup>
```

5 PACKAGE HEADER

5.1 OVERVIEW

The ***Package Header*** (packageHeader element of packageHeaderType) is an XML Complex Type that contains metadata that apply to the whole XFDU Package. This metadata may include data to inform the XFDU parsing software about volume metadata (e.g., logical volume information and specification version), administrative metadata (e.g., author and creation data) and technical data (e.g., hardware and operating system).

The package header type has two elements:

- environmentInfo – contains application specific information either defined by an extension of the XFDU Schema or by freeform XML;
- volumeInfo – contains XFDU volume related metadata such as XFDU specification version and logical volume sequence information.

The package header type has a single attribute, ID: an XML ID.

5.2 XML SCHEMA packageHeader Type

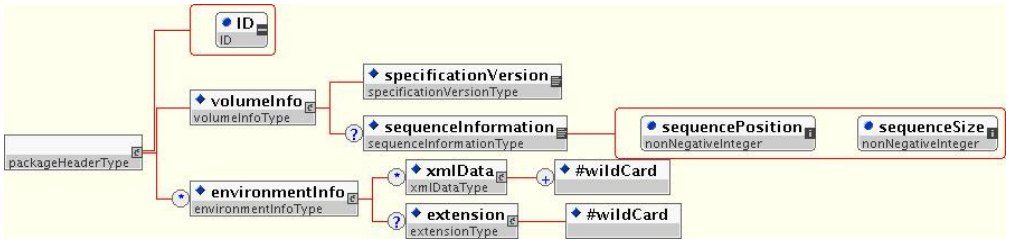


Figure 5-1: packageHeaderType Schema Diagram

```
<xsd:complexType name="packageHeaderType">
    <xsd:annotation>
      <xsd:documentation>packageHeaderType: Complex Type for metadata about the
      mapping of the logical packages to the physical structures. The
      package header type has two elements:
        -volumeInfo – contains  XFDU volume related metadata (.i.e., XFDU specification version
          and sequence information
        - environmentInfo – contains application specific information either defined by an extension of the XFDU
          Schema or by freeform XML.
          packageHeaderType has a single attribute, ID: an XML ID.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="volumeInfo" type="xfdu:volumeInfoType"/>
      <xsd:element name="environmentInfo" type="xfdu:environmentInfoType" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ID" type="xsd:ID" use="required"/>
</xsd:complexType>
<xsd:complexType name="volumeInfoType">
```

```
    <xsd:annotation>
      <xsd:documentation>
        Contains XFDU software related system information, including one mandatory element
          - specificationVersion, which specifies the version of the XFDU specification to which this manifest complies.
        Additionally it has one optional element-sequenceInformation that holds the information about sequence
        of XFDUs and the position of the current one in it.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="specificationVersion" type="xfdu:specificationVersionType" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="sequenceInformation" type="xfdu:sequenceInformationType" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="environmentInfoType">
    <xsd:annotation>
      <xsd:documentation>
        Environment info provides meta information related to the environment where the XFDU was created.
        Since environment information may be specific to a concrete XFDU producer, environment information can have
        only two optional elements:
        -xmlData - can hold application specific information.
        -extension -wild card that serves as extension point for other namespaces
      </xsd:documentation>
    </xsd:annotation>
      <xsd:sequence>
        <xsd:element name="xmlData" type="xfdu:xmlDataType" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="extension" type="xfdu:extensionType" minOccurs="0"/>
      </xsd:sequence>
  </xsd:complexType>

  <xsd:simpleType name="specificationVersionType">
    <xsd:annotation>
      <xsd:documentation>
        An entity of this type is used to indicated CCSDS-bound version of XFDU specification
      </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
  <xsd:complexType name="sequenceInformationType">
    <xsd:annotation>
      <xsd:documentation>
        An element of this type encapsulates information about the position of the encapsulating XFDU
        package In a sequence of physical XFDU packages that form the identified logical XFDU unit.
        The sequenceInformation element is a string that acts as an identifier for the logical XFDU.
        SequenceInformationType has two mandatory attributes:
        1. sequencePosition - the position of this XFDU package in the sequence; if 0 is specified
        and sequenceSize is unknown, it means that it is last in the sequence
        2. sequenceSize - the total number of packages in the sequence; if its value is 0 this means
        size is unknown
      </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
      <xsd:extension base = "xsd:string">
        <xsd:attribute name = "sequencePosition" type = "xsd:nonNegativeInteger" use="required"/>
        <xsd:attribute name = "sequenceSize" type = "xsd:nonNegativeInteger" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
```

5.3 EXAMPLES

5.3.1 PACKAGE HEADER

In the following example, package header has two elements - volume info and environment info.

Volume info has two elements:

- specification version –specifies the value of XFDU specification to which this XFDU adheres;
- sequence information - specifies that this XFDU is part of sequence of 10 XFDUs and it is first in the sequences XFDU.

Environment info has two elements:

- xml data – encapsulates information about platform where this XFDU has been created;
- cip identification information – encapsulates an application specific information (CIP) that is beyond standard XFDU specification; this shows how XFDU producers can extend XFDU with information specific to their application.

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdu:XFDU xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:ccsds:schema:xfdu:1 http://www.ccsds.org/xfdu/xfdu-1.0.xsd"
   xmlns:xfdu="urn:ccsds:schema:xfdu:1">

   <packageHeader ID="packageHeader">
      <volumeInfo>
         <specificationVersion>1.0</specificationVersion>
         <!-- sequence information attribute is specified by producer-->
         <sequenceInformation sequenceSize="10"
sequencePosition="1">producer1.seq10</sequenceInformation>
      </volumeInfo>
      <environmentInfo>
         <xmlData>
            <platform>Linux2.4.22-1.2129.nptl</platform>
         </xmlData>
         <extension>
            <cipIdentificationInformation xmlns="cip"
               producer_ID="NSSDC"
               project_ID="NSSDC_SPMS"
               cip_ID="NSSDC_SPMS-00216_PKG01"
               cip_template_ID="SPMS-00216-DO"
               cip_template_Location="complexTemplateCIP.xml"/>
            </extension>

      </environmentInfocipcipcipcipcipCIP
```

6 CONTENT UNIT

6.1 OVERVIEW

A ***Content Unit*** (contentUnit element of contentUnitType) is the basic structural unit of the XFDU. Content Unit elements may include other Content Units, may be internal pointers to elements in the Data Object section or may be external pointers to other XFDUs. Therefore a Content Unit can be used to associate a Data Object with one or more Metadata Objects, and multiple Content Units can present a hierarchical view of these data/metadata associations.

The Content Unit attributes allow reference to associated Metadata Objects by internal pointers to elements in the Metadata Object section. Several of these attributes may be used to categorize the referenced Metadata Object distinguishing among Representation Information, Preservation Description Information (PDI), and Descriptive Information as defined in the OAIS reference model.

6.2 XML SCHEMA FOR contentUnitType

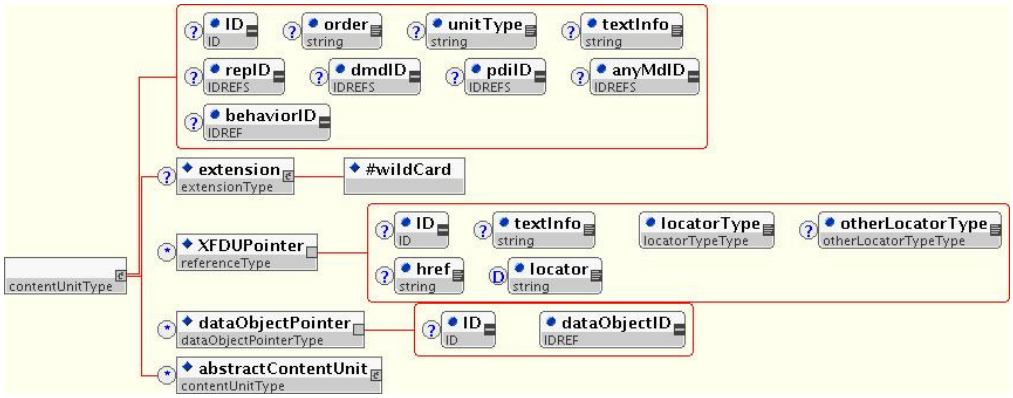


Figure 6-1: contentUnitType Schema Diagram

```
<xsd:complexType name="contentUnitType">
<xsd:annotation>
  <xsd:documentation>ContentUnit Complex Type The XFDU standard
  represents a data package structurally as a series of nested
  content units, that is, as a hierarchy (e.g., a data product,
  which is composed of datasets, which are composed of time
  series, which are composed of records). Every content node in
  the structural map hierarchy may be connected (via subsidiary
  XFDUPointer or dataObjectPointer elements) to information objects which
  represent that unit as a portion of the whole package. The contentUnitType
  is declared as a base type for concrete implementations of contentUnit;
  The content unit element has the following attributes:
  1.ID (an XML ID);
  2.order: a numeric string (e.g., 1.1, 1.2.1, 3,) representation
  of this unit's order among its siblings (e.g., its sequence); order attribute is not meant to be used
```

```
      for processing purposes. It is here only for visualization purposes of the potential reader of XML instance.
      It is not guaranteed that any software will take value of order attribute into account. contentUnit nesting is
      the primary means for determining order and level of the content information.
      3.textInfo: a string label to describe this contentUnit to an end
      user viewing the document, as per a table of contents entry
      4.repID: a set of IDREFs to representation information sections
      within this XFDU document applicable to this contentUnit.
      5.dmdID: a set of IDREFS to descriptive information sections
        within this XFDU document applicable to this contentUnit.
      6.pdiID: a set of IDREFS to preservation description information
        sections within this XFDU document applicable to this
        contentUnit
      7.anyMdID: a set of IDREFS to any other metadata sections that do not fit
        rep,dmd or pdi meatdata related to this  contentUnit
      8.unitType: a type of content unit (e.g., Application
        Data Unit, Data Description Unit, Software Installation Unit, etc.).
      9. behaviorID-an XML ID reference pointing to associate behavior.

      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="extension" type="xfdu:extensionType" minOccurs="0"/>
      <xsd:element name="XFDUPointer" type="xfdu:referenceType" minOccurs="0" maxOccurs="unbounded">
        <xsd:annotation>
          <xsd:documentation>XFDUPointer:XFDU Pointer. The XFDUPointer element allows a
          content unit to be associated with a separate XFDU containing
          the content corresponding with that contentUnit, rather than
          pointing to one or more internal dataObjects. A typical instance of
          this would be the case of a thematic data product that collects
          data products from several instruments observe an event of
          interest. The content units for each instrument datasets might
          point to separate XFDUs, rather than having dataObjects and dataObject
          groups for every dataset encoded in one package. The XFDUPointer
          element may have the following attributes:
          1. ID: an XML ID for this element;
          2. locatorType: the type of location contained in the href attribute;
          3. otherLocatorType: a string to indicate an alternative locator type
             if the locatorType attribute itself has a value of "OTHER."
          NOTE: The XFDUPointer is an empty element. The location of the resource pointed to
          Must be stored in the href attribute.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="dataObjectPointer" type="xfdu:dataObjectPointerType" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="xfdu:abstractContentUnit" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ID" type="xsd:ID" use="optional"/>
    <xsd:attribute name="order" type="xsd:string"/>
    <xsd:attribute name="unitType" type="xsd:string"/>
    <xsd:attribute name="textInfo" type="xsd:string"/>
    <xsd:attribute name="repID" type="xsd:IDREFS"/>
    <xsd:attribute name="dmdID" type="xsd:IDREFS"/>
    <xsd:attribute name="pdiID" type="xsd:IDREFS"/>
    <xsd:attribute name="anyMdID" type="xsd:IDREFS"/>
    <xsd:attribute name="behaviorID" type="xsd:IDREF"/>
  </xsd:complexType>
  <xsd:element name="abstractContentUnit" type="xfdu:contentUnitType" abstract="true">
    <xsd:annotation>
      <xsd:documentation>abstractContentUnit is abstract implementation of
      contentUnitType. It cannot be instantiated in the instance
      document. Instead, concrete implementations would have to be
      used which are declared part of contentUnit substitutionGroup
      </xsd:documentation>
```

```
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="contentUnit" type="xfdu:contentUnitType"
substitutionGroup="xfdu:abstractContentUnit">
    <xsd:annotation>
      <xsd:documentation>contentUnit is a basic concrete
        implementation of an abstract contentUnit. Its instance can be used
        in the instance document in the place where contentUnit declared
        to be present.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
```

6.3 EXAMPLES

6.3.1 SIMPLE CONTENT UNIT

In the following example, the content unit points to several types of metadata:

- representation metadata with XML ID 'atdMD'; this metadata is categorized as representation metadata; thus, it is referred to by placing its XML ID as value of repID attribute;
- preservation metadata with XML ID 'provenance', this metadata is categorized as preservation metadata; thus, it is referred to by placing its XML ID as value of pdiID attribute;
- description metadata with XML ID ECSDMD, this metadata is categorized as descriptive metadata; thus, it is referred to by placing its XML ID as value of pdiID attribute.

```
<contentUnit repID = "atdMD" pdiID = "provenance" dmdID = "ECSDMD">
  <dataObjectPointer dataObjectID = "mpeg21"/>
</contentUnit>
```

6.3.2 INFORMATION PACKAGE CONTENT UNIT

The following example demonstrates how different types of content units may be used via the content unit substitution group provided by the XFDU schema. The following content unit is an extension of the contentUnitType and part of the abstractContentUnit substitution group. In this example an archive has specialized the contentUnit schema and added standard elements for describing the generic designs and relationships for submissions to that archive.

```
<informationPackageMap xmlns="">
  <cip:cipContentUnit xmlns:cip="cip">
    <xfdu:contentUnit ID="NSSDC_SPMS-00216_PKG01" anyMdID="SPMS-00216-DO">
      <xfdu:contentUnit ID="DATA" anyMdID="SPMS-00216-DO">
        <xfdu:contentUnit ID="DATA01" dmdID="METADATA01" anyMdID="SPMS-00216-DO">
          <dataObjectPointer dataObjectID="DATA_FILE01" />
        </xfdu:contentUnit>
        <xfdu:contentUnit ID="DATA02" dmdID="METADATA02" anyMdID="SPMS-00216-DO">
          <dataObjectPointer dataObjectID="DATA_FILE02" />
```

```
            </xfdu:contentUnit>
            <xfdu:contentUnit ID="DATA03" dmdID="METADATA03" anyMdID="SPMS-00216-DO">
              <dataObjectPointer dataObjectID="DATA_FILE03" />
            </xfdu:contentUnit>
            <xfdu:contentUnit ID="DATA04" dmdID="METADATA04" anyMdID="SPMS-00216-DO">
              <dataObjectPointer dataObjectID="DATA_FILE04" />
            </xfdu:contentUnit>
          </xfdu:contentUnit>
        </xfdu:contentUnit>
        <!-- cipDescriptor and cipTemplate are part of specialized cipContentUnit -->
        <cip:cipDescriptor locType="URL" xlink:href="SPMS_00216.xml" />
        <cip:cipTemplate locType="URL" xlink:href="complexTemplateCIP.xml" />
    </cip:cipContentUnit>
</informationPackageMap>


<!-- Corresponding schema fragment-->
<xsd:element name="cipContentUnit" type="cipContentUnitType" substitutionGroup="xfdu:abstractContentUnit"/>
  <xsd:complexType name="cipContentUnitType">
    <xsd:complexContent>
      <xsd:extension base="xfdu:contentUnitType">
        <xsd:sequence>
          <xsd:element name="cipDescriptor" type="cipDescriptorLocationType"/>
          <xsd:element name="cipTemplate" type="cipTemplateLocationType"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="cipDescriptorLocationType">
    <xsd:complexContent>
      <xsd:extension base="xfdu:referenceType"/>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="cipTemplateLocationType">
    <xsd:complexContent>
      <xsd:extension base="xfdu:referenceType"/>
    </xsd:complexContent>
  </xsd:complexType>
```

6.4 SEMANTICS

6.4.1 CONTENT UNIT TYPES

In this version of the Recommended Standard , the unitType attribute of the Content Unit is allowed to be free text with the suggested values and the semantics for each unit type being discussed in the XFDU Green Book. It is anticipated that in future issues of this Recommended Standard this list may become an enumerated type and the semantics of each Content Unit type may be specified and enforced by Schematron-like mechanisms.

6.4.2 XFDU AS A CONTENT UNIT

The xfduPointer element allows any contentUnit to reference an external XFDU. In this Recommendation the semantics of an XFDU being referenced within from within a contentUnit is that the internal structure of the referenced XFDU is not visible through the containing content Unit. In other words there is no implication that the hierarchical structure

of the referencing XFDU is extended by the hierarchical structure of the refererenced XFDU. Future versions of this Recommendation may provide this capability but the semantics are not well understood and there were no current requirements for this construct.

7 INFORMATION PACKAGE MAP

7.1 OVERVIEW

The ***Information Package Map*** (informationPackageMap element of informationPackageMapType) outlines a hierarchical structure for the collection of content objects being packaged, by a series of nested contentUnit elements. The Information Package Map is the highest level Content Unit of the nested Content Units in XFDU. The order and the nesting of the contained Content Units provide the Information Model for the XFDU and should provide an access path to all the data and metadata objects within the XFDU.

The Information Package Map provides attributes for identifying, classifying, and describing itself.

7.2 XML SCHEMA informationPackageMapType

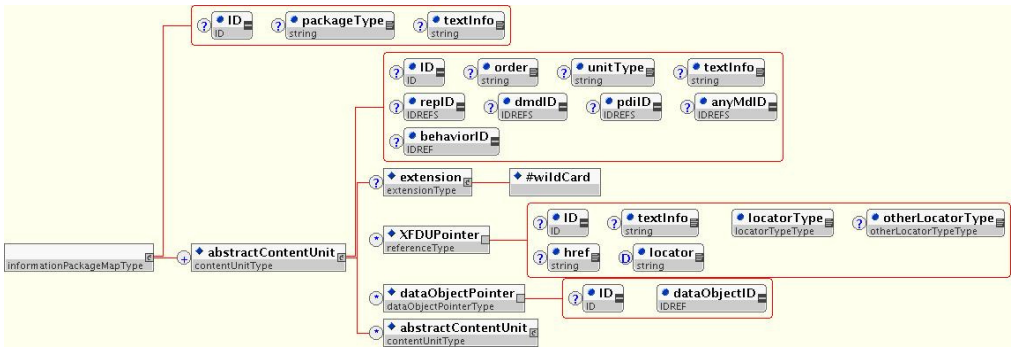


Figure 7-1: InformationPackageMapType Schema Diagram

```
<xsd:complexType name="informationPackageMapType">
  <xsd:annotation>
    <xsd:documentation>informationPackageMapType Complex Type The Information Package Map
      outlines a hierarchical structure for the original object being encoded, using a series of nested contentUnit elements.
        An element of informationPackageMapType has the following attributes:
      1. ID: an XML ID for the element;
      2. packageType: a type for the object. Typical values will be "AIP" for a map which describes
        a complete AIP obeying all constraints and cardinalities in the OAIS reference model.
        "SIP" for a map which describes a Submission Information Package.
      3. textInfo: a string to describe the informationPackageMap to users.
      4. anyAttribute - wild-carded attribute extension point
        Concrete implementations of abstractContentUnit (contentUnit etc.) must be used in the instance document.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element ref="xfdu:abstractContentUnit" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="ID" type="xsd:ID" use="optional"/>
  <xsd:attribute name="packageType" type="xsd:string"/>
  <xsd:attribute name="textInfo" type="xsd:string"/>
  <xsd:anyAttribute namespace="##other" processContents="lax"/>
</xsd:complexType>
```

7.3 EXAMPLES

7.3.1 AN INFORMATION PACKAGE MAP

The following example demonstrates information package map with both nested and non-nested content units. It has two top level content units representing two data objects. Data objects are pointed to via dataObjectPointer element. The first content unit relates its data object to representation, provenance and descriptive metadata via repID, pdiID and dmdID attributes respectively. The first content unit has number of nested content units each of which represents its data object and associated metadata.

```
<informationPackageMap ID="informationPackageMap">
  <xfdu:contentUnit ID="cu1" repID = "atdMD" pdiID = "provenance" dmdID = "ECSDMD">
    <dataObjectPointer dataObjectID = "mpeg21"/>
    <xfdu:contentUnit ID="cu2" order = "1" textInfo = "Root content unit for HDF data">
      <xfdu:contentUnit ID="cu3" order = "1.1" pdiID = "provenance" textInfo = "content unit for hdfFile0"dmdID = "ECSDMD">
        <dataObjectPointer dataObjectID = "hdfFile0"/>
      </xfdu:contentUnit>
      <xfdu:contentUnit ID="cu4" order = "1.2" pdiID = "provenance" textInfo = "content unit for hdfFile1" dmdID = "ECSDMD">
        <dataObjectPointer dataObjectID = "hdfFile1"/>
      </xfdu:contentUnit>
      <xfdu:contentUnit ID="cu5" order = "1.3" pdiID = "provenance" textInfo = "content unit for hdfFile2"dmdID = "ECSDMD">
        <dataObjectPointer dataObjectID = "hdfFile2"/>
      </xfdu:contentUnit>
    </xfdu:contentUnit>
    <xfdu:contentUnit ID="cu6" textInfo = "content unit for orbit data">
      <dataObjectPointer dataObjectID = "orbitalData"/>
    </xfdu:contentUnit>
  </xfdu:contentUnit>
  <xfdu:contentUnit ID="cu7" textInfo = "content unit ATD metadata">
    <dataObjectPointer dataObjectID = "ATDMD"/>
  </xfdu:contentUnit>
</informationPackageMap>
```

7.4 SEMANTICS

Future issues of this Recommendation may allow multiple Information Package Maps in an XFDU. The semantics of multiple Information Package Maps in an XFDU are not well understood and there were no requirements for this construct.

8 DATA OBJECTS

8.1 OVERVIEW

The ***Data Object Section*** (dataObjectSection element of dataObjectSectionType) contains one or more ***Data Object*** elements (dataObject element of dataObjectType). Each Data Object contains one or more ***Byte Stream*** (byteStream) elements that reference or contain the current digital object content and any number of optional Transformation Objects(transformObject) that contain required information (e.g., algorithms and parameters) to reverse any transformations to the digital content and restore them to the original binary data object.

The dataObject element provides access to the current content files for a XFDU document.

8.2 XML SCHEMA FOR DATA OBJECT TYPE

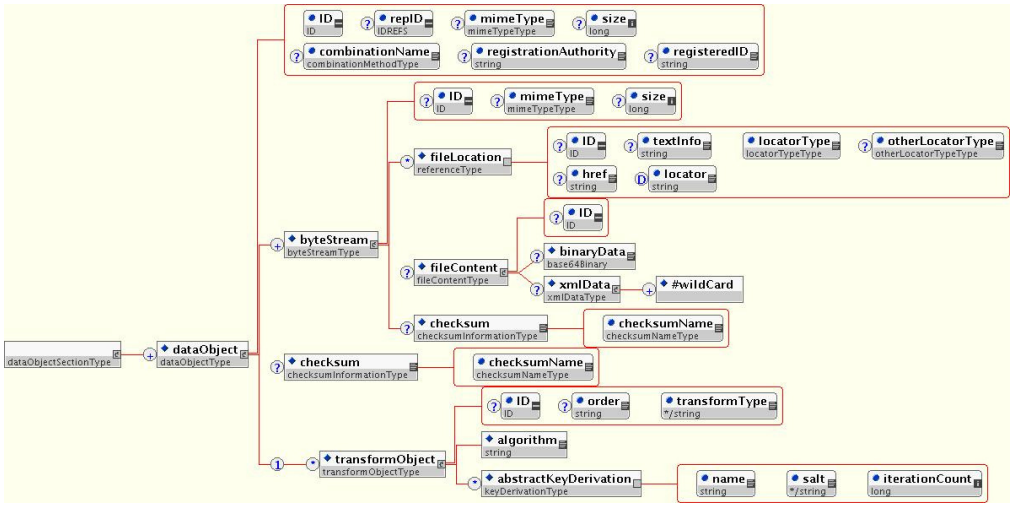


Figure 8-1: dataObjectType Schema Diagram

```
<xsd:complexType name="keyDerivationType">
  <xsd:annotation>
    <xsd:documentation>keyDerivationType contains the information
    that was used to derive the encryption key for this dataObject.
    Key derivation type contains:
     name - name of algorithm used
    salt - 16-byte random seed used for that algorithm initialization
    iterationCount - number of iterations used by the algorithm to derive the key
    </xsd:documentation>
  </xsd:annotation>
  <xsd:attribute name="name" use="required" type="xsd:string"/>
  <xsd:attribute name="salt" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:length value="16"/>
```

```
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="iterationCount" use="required" type="xsd:long"/>
  </xsd:complexType>
  <xsd:element name="abstractKeyDerivation" type="xfdu:keyDerivationType" abstract="true">
    <xsd:annotation>
      <xsd:documentation>
      abstractKeyDerivation is declared abstract
      so that it can be used for element substitution in cases when the default keyDerivation is not
      sufficient. In order for creating more specific key derivation constructs, one would have to
      extend from keyDerivationType to a concrete type, and then create an element of that new type. Finally,
      in an instance of XML governed by this schema, the reference to keyDerivation in an instance of
      transformObject element would point not to an instance of keyDerivation element, but rather to an instance of the
      custom element. In other words, keyDerivation would be SUBSTITUTED with a concrete key derivation element.
      In cases where default functionality is sufficient, the provided defaultKeyDerivation element can be used for the
      substitution.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="keyDerivation" type="xfdu:keyDerivationType" substitutionGroup="xfdu:abstractKeyDerivation">
    <xsd:annotation>
      <xsd:documentation>
        Default implementation of key derivation type.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:complexType name="transformObjectType">
    <xsd:annotation>
      <xsd:documentation>transformObjectType: transformation information: An element
      of transformObjectType contains all of the information required to reverse the
      transformations applied to the original contents of the dataObject. It
      has two possible subsidiary elements: The algorithm element
      contains information about the algorithm used to encrypt the
      data. The key-derivation element contains the information that
      was used to derive the encryption key for this dataObject It has three
      attributes:
        1. ID: an XML ID
        2. transformType: one of n predefined transformations types.
           Current valid types are compression,
           encryption, authentication.
        3. order: If there are more than one transformation elements in an dataObject
           this integer indicates the order in which the reversal transformations should be applied.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="algorithm" type="xsd:string">
        <xsd:annotation>
          <xsd:documentation>algorithm element contains information
            about the algorithm used to encrypt the data.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element ref="xfdu:abstractKeyDerivation" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ID" type="xsd:ID"/>
    <xsd:attribute name="order" type="xsd:string"/>
    <xsd:attribute name="transformType" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="COMPRESSION"/>
          <xsd:enumeration value="AUTHENTICATION"/>
          <xsd:enumeration value="ENCRYPTION"/>
```

```
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
  <xsd:complexType name="byteStreamType">
    <xsd:annotation>
      <xsd:documentation>byteStreamType: An element of  byteStreamType
      provides access to the current content of dataObjects for a XFDU
      document. The byteStreamType: has the following four attributes: ID (an XML ID);
      mimeType: the MIME type for the dataObject; size: the size of the dataObject
      in bytes.
      Checksum information provided via option checksum element.
      The data contained in these attributes is relevant to the final state of data object
      after all possible transformations of the original data.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="fileLocation" type="xfdu:referenceType" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="fileContent" type="xfdu:fileContentType" minOccurs="0"/>
      <xsd:element name="checksum" type="xfdu:checksumInformationType" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="ID" use="optional" type="xsd:ID"/>
    <xsd:attribute name="mimeType" type="xfdu:mimeTypeType"/>
    <xsd:attribute name="size" type="xsd:long"/>
  </xsd:complexType>
  <xsd:complexType name="dataObjectType">
    <xsd:annotation>
      <xsd:documentation>dataObjectType: An element of dataObjectType
      contains current byteStream content and any required data to restore
      them to the form intended for the original designated community.
      It has two possible subsidiary elements: The byteStream element
      provides access to the current content dataObjects for an XFDU
      document. An element of dataObjectType must contain 1 or many byteStream elements
      that may contain a fileLocation element, which provides a pointer to
      a content byteStream, and/or a fileContent element, which wraps an
      encoded version of the dataObject. An element of dataObjectType may contain one or
      more transformation elements that contain all of the
      information required to reverse each transformation applied to
      the dataObject and return the original binary data object.
      The dataObjectType has the following attributes:
      1. ID: an XML ID
      2. mimeType: the MIME type for the dataObject
      3. size: the size of the dataObject in bytes
      4. checksum: a checksum for dataObject. Checksum information provided via optional checksum element.
      5. repID list of representation metadata IDREFs.
      NB: The size, checksum, and mime type are related to the original data before any transformations occurred.
      6. combinationName - specifies how multiple byteStream objects in a single dataObject should be concatenated
      7. registrationGroup attribute group that provides registration information
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="byteStream" type="xfdu:byteStreamType" minOccurs="1"
maxOccurs="unbounded"/>
      <xsd:element name="checksum" type="xfdu:checksumInformationType" minOccurs="0" maxOccurs="1"/>
      <xsd:sequence>
        <xsd:element name="transformObject" type="xfdu:transformObjectType" minOccurs="0"
          maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:sequence>
    <xsd:attribute name="ID" type="xsd:ID" use="required"/>
    <xsd:attribute name="repID" type="xsd:IDREFS"/>
    <xsd:attribute name="mimeType" type="xfdu:mimeTypeType"/>
    <xsd:attribute name="size" type="xsd:long"/>
```

```
    <xsd:attribute name="combinationName" type="xfdu:combinationMethodType" use="optional"/>
    <xsd:attributeGroup ref="xfdu:registrationGroup"/>
  </xsd:complexType>
  <xsd:complexType name="dataObjectSectionType">
    <xsd:annotation>
      <xsd:documentation>dataObjectSectionType : a container for one or more elements of dataObjectType
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="dataObject" type="xfdu:dataObjectType" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:simpleType name="combinationMethodType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="concat"/>
    </xsd:restriction>
  </xsd:simpleType>
```

8.3 EXAMPLES

8.3.1 VERIFY THE CHECKSUM OF THE FILE

Reader of the document can verify checksum of animation file by comparing its checksum with checksum value specified in checksum element of dataObject element.

```
<dataObject repID = "mathMLAlgRepMD" ID = "mpeg21">
  <byteStream mimeType = "video/mpeg" ID = "mpeg21AnimData" size = "414131">
    <fileLocation locatorType = "URL" href = "file:packagesamples/scenario1/mpeg21.mpg"/>
    <checksum checksumName="CRC32">b3eb4b34</checksum>
  </byteStream>
</dataObject>
```

8.3.2 SPECIFICATION OF MIMETYPE AND CHECKSUM WITH TRANSFORMATIONS

Mime type and checksum are specified at two levels. The values of the mimeType attribute and checksum element of the Data Object specify the mime type and checksum of the original data object, (i.e.,byte stream before any transformations were applied). The values of the mimeType and checksum attributes of the byteString object are those of the received data object before any transformations are reversed (i.e., this encoded byte stream).

```
<dataObject size = "151672" mimeType = "application/pdf" ID = "ATDMD">
  <byteStream mimeType = "application/octetstream" ID = "atdMDbs" size = "110874">
    <fileLocation locatorType = "URL" href = "file:packagesamples/scenario1/atd.pdf" />
    <checksum checksumName="CRC32">ad78ad5d</checksum>
  </byteStream>
  <checksum checksumName="CRC32">6d0e30ea</checksum>
  <transformObject transformType = "ENCRYPTION">
    <algorithm>blowfish</algorithm>
  </transformObject>
</dataObject>
```

8.3.3 REFERENCING AND INCLUSION OF DATA CONTENT

The data object that existed at the moment of packaging is base64 encoded within afileContent element and included physically in the Manifest. The fileLocation element specifies an HTTP GET URL to request the latest version of data from an online registry/repository.

```
<dataObject mimeType = "application/octetstream" ID = "orbitalData">
  <byteStream ID = "orbitData">
    <fileLocation locatorType = "URL" href = "http://coin.gsfc.nasa.gov:8080/ims-bin/3.0.1/nph-ims.cgi?
msubmit=yes&amp;lastmode=SRCHFORM"/>
<fileContent>
<binaryData>UEsDBBQACAAIAKqMBC8AAAAAAAAAAAAAAAPAAAAeGZkdS8uY2xhc3NwYXRotZXfS8MwEMff/StK35OugqCw
H4hO0I...
</binaryData>
    </fileContent>
    <checksum checksumName="CRC32">b3eb4b34</checksum>
  </byteStream>
</dataObject>
```

8.4 SEMANTICS

These are the semantics for the existence of:

a) Multiple fileLocation or fileContent elements in a single byteStream:

   1) One fileContent and one fileLocation means the fileContent should serve as backup if the fileLocation is not accessible.

   2) One fileLocation referencing an object in the XFDU Package and one fileLocation accessing an object outside the XFDU Package means the object located within the package should serve as backup if the remote fileLocation is not accessible.

   3) Other forms of multiple fileLocations are undefined at this time. Implementers are cautioned against defining semantics for this construct.

   4) One fileContent and multiple fileLocation semantics are undefined at this time. Implementers are cautioned against defining semantics for this construct.

b) Multiple byteStreams in a single dataObject:

   1) The combinationName attribute in the data object specifies how the byteStreams are to be combined. Currently the only allowed value of the combinationName attribute is “concat”. This indicates the multiple byteStreams (i.e., multiple physical files) must be concatenated to form a single described object (e.g., an image).

   2) It is anticipated that in future issues of this Recommendation this list of combination will include other methods for combining multiple byteStreams into a single described object.

c) Multiple byteStreams, combined into a single byteStream by the use of a transformation algorithm (e.g., ZIP or TAR):

   1) Each contained byteStreams will be treated as if they were each contained in a single dataObject.

   2) Any additional transformations specified in the original data Object will be performed on each byteStream independently.

9 METADATA OBJECTS

9.1 OVERVIEW

The ***Metadata Section*** (metadataSection element of metadataSectionType) contains zero or more ***Metadata Object*** elements (metadataObject element of metadataObjectType) that record all of the static metadata for all entities in the XFDU package. The metadata schema allows the package designer to define any metadata model by providing attributes for both the metadata categories discussed in section 6 Content Units and a classification scheme for finer definition within categories. The XFDU Manifest Schema provides predefined metadata categories and classes via enumerated attributes that follow the OAIS information model as follows:

- Descriptive Information, intended for the use of Finding Aids such as Catalogs or Search Engines, may be categorized as 'DMD' and further classified as 'DESCRIPTION' or 'OTHER'.
- Representation Information may be categorized as 'REP' and then further classified as 'SYNTAX', 'DED' (data entity dictionary), or 'OTHER'.
- Preservation Description Information may be categorized as 'PDI' and then further classified as 'REFERENCE', 'CONTEXT', 'PROVENANCE', 'FIXITY' or 'OTHER'.

The elements describing Metadata Objects are used to either encapsulate the actual object in base64 or XML, to point to a file either within the XFDU Package or contained in an external resource, or to point to a Data Object in the Data Object section. This allows a Metadata Object to also be described as Data Object in the Data Objects section. Since the dataObject includes an attribute that is an internal pointer to Representation Information, a Metadata Object can be associated with its own Representation Information. Note that this mechanism allows the construction of OAIS defined 'Representation Nets' when the associated Representation Metadata Objects are also held as Data Objects. The attributes of a Metadata Object, like those of a Content Unit, can be used to categorize and classify the objects, including the ability to distinguish among Representation Information, Preservation Description Information (PDI), and Descriptive Information as shown in figure **Error! Reference source not found.**

Deleted: Error! Reference source not found.

9.2 XML SCHEMA FOR METADATA OBJECTS

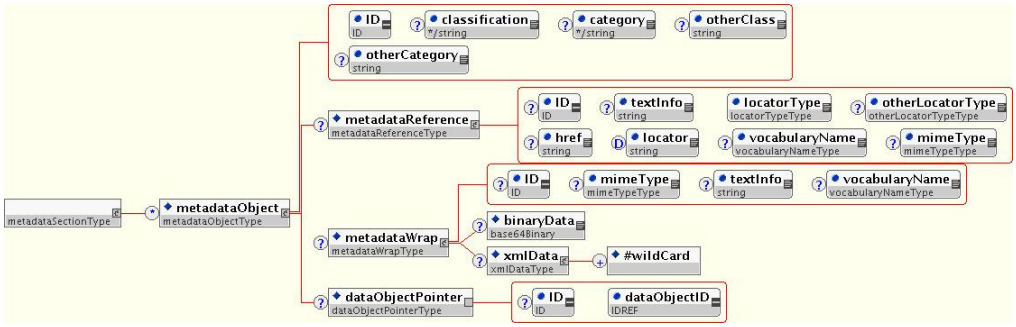


Figure 9-1: metadataObjectType and metadataSectionType Schema Diagram

```
<xsd:complexType name="metadataObjectType">
  <xsd:annotation>
    <xsd:documentation>metadataObjectType Complex Type A generic
    framework for pointing to/including metadata within a XFDU
    document, a la Warwick Framework. A metadataObject element may have the
    following attributes:
    1. ID: an XML ID for this element.
    2. classification - concrete type of metadata represented by this element of metadataObjectType
    3. category - type of metadata class to which this metadata belongs (e.g., DMD.REP, etc.)
    4. otherClass - type of metadata in case classification contains value of "OTHER"
    5. otherCategory - type of metadata class in case category contains value of "OTHER"
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="metadataReference" type="xfdu:metadataReferenceType" minOccurs="0"/>
    <xsd:element name="metadataWrap" type="xfdu:metadataWrapType" minOccurs="0"/>
    <xsd:element name="dataObjectPointer" type="xfdu:dataObjectPointerType" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="ID" use="required" type="xsd:ID"/>
  <xsd:attribute name="classification">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="DED"/>
        <xsd:enumeration value="SYNTAX"/>
        <xsd:enumeration value="FIXITY"/>
        <xsd:enumeration value="PROVENANCE"/>
        <xsd:enumeration value="CONTEXT"/>
        <xsd:enumeration value="REFERENCE"/>
        <xsd:enumeration value="DESCRIPTION"/>
        <xsd:enumeration value="OTHER"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="category">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="REP"/>
        <xsd:enumeration value="PDI"/>
        <xsd:enumeration value="DMD"/>
        <xsd:enumeration value="OTHER"/>
        <xsd:enumeration value="ANY"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="otherClass" type="xsd:string"/>
```

```
  <xsd:attribute name="otherCategory" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="metadataReferenceType">
  <xsd:annotation>
    <xsd:documentation>metadataReferenceType: metadata reference.
    An element of metadataReferenceType is a
    generic element used throughout the XFDU schema to provide a
    pointer to metadata which resides outside the XFDU document.
    MetadataReferenceType has the following attributes:
    1. ID: an XML ID;
    2. locatorType: the type of locator contained in the body of the element;
    3. otherLocatorType: a string indicating an alternative type of locator when
      the locatorType attribute value is set to "OTHER.";
    4. href: actual location (e.g., URL)
    5. mimeType: the MIME type for the metadata being pointed at;
    6. vocabularyName: the name of the well known standard vocabulary (e.g., Dublin Core)
      of the metadata being pointed at;
    7. textInfo: a label to display to the viewer of the XFDU document identifying the metadata
    NB: metadataReference is an empty element. The location of the
    metadata must be recorded in the href attribute.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="xfdu:referenceType">
      <xsd:attribute name="vocabularyName" type="xfdu:vocabularyNameType"/>
      <xsd:attribute name="mimeType" type="xfdu:mimeTypeType"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="metadataWrapType">
  <xsd:annotation>
    <xsd:documentation>metadataWrapType: metadata wrapper. An element of metadataWrapType is a
    generic element used throughout the XFDU schema to allow the
    encoder to place arbitrary metadata conforming to other
    standards/schema within a XFDU document.  The metadataWrapType
    can have the following attributes:
    1. ID: an XML ID for this element;
    2. mimeType: the MIME type for the metadata contained in the element;
    3. vocabularyName: the type of metadata contained (e.g., MARC, EAD, etc.);
    4. textInfo: a label to display to the viewer of the XFDU document identifying the metadata.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="xfdu:fileContentType">
      <xsd:attribute name="mimeType" type="xfdu:mimeTypeType"/>
      <xsd:attribute name="textInfo" type="xsd:string"/>
      <xsd:attribute name="vocabularyName" type="xfdu:vocabularyNameType"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:simpleType name="vocabularyNameType">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
<xsd:complexType name="metadataSectionType">
  <xsd:sequence>
    <xsd:element name="metadataObject" type="xfdu:metadataObjectType" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

9.3 METADATA OBJECT EXAMPLES

9.3.1 METADATA SECTION USING OAIS INFORMATION MODEL

In this example, several metadata objects are presented that show usage of different categories and classes:

- Metadata object with XML ID 'ECSDMD' is categorized as descriptive metadata of class OTHER; thus, its category attribute has a value of DMD and classification attribute has a value of OTHER.
- Metadata object with XML ID 'provenance' is categorized as preservation metadata of class PROVENANCE; thus, its category attribute has a value of PDI and classification attribute has a value of PROVENANCE.
- Metadata object with XML ID 'atdMD' is categorized as representation metadata of class OTHER; thus, its category attribute has a value of REP and classification attribute has a value of OTHER.

```
<metadataSection>
   <metadataObject ID = "ECSDMD" classification = "OTHER" category = "DMD">
      <metadataReference vocabularyName="OTHER" mimeType = "text/xml" textInfo = "spacecraft description" locatorType = "URL"
href = "file:packagesamples/scenario1/ecsdmd.xml"/>
   </metadataObject>
   <metadataObject ID = "provenance" classification = "PROVENANCE" category = "PDI">
      <metadataReference vocabularyName = "OTHER" mimeType = "text/xml" textInfo = "processing  history  XML file" locatorType =
"URL" href = "file:packagesamples/scenario1/pdi.xml" />
   </metadataObject>
   <metadataObject ID = "atdMD" classification = "OTHER" category = "REP">
      <dataObjectPointer dataObjectID = "ATDMD"/>
   </metadataObject>
</metadataSection>
```

10 BEHAVIOR SECTION AND BEHAVIOR OBJECTS

10.1 OVERVIEW

The Behavior Section (behaviorSection element of behaviorSectionType) contains zero or more Behavior Object elements (behaviorObject element of behaviorObjectType) that associate executable behaviors with content in the XFDU object. A Behavior Object contains an Interface Definition element (interfaceDefinition element of interfaceDefinitionType) that represents an abstract definition of the set of behaviors represented by a particular Behavior Object. A Behavior Object also may contain zero or more Mechanism elements that are modules of executable code that implement and run the behaviors defined abstractly by the interface definition. In the current XML schema a Mechanism is represented by the element abstractMechanism. An abstract element cannot be instantiated in the instance document. Instead, concrete implementations based on techniques such as JAVA, WSDL or Ant would be declared as part of the mechanism substitution group and be used. Behavior Objects may be nested to indicate chaining of execution. There are currently no concrete implementations of mechanism objects but the element abstractMechanism and the related substitution group have been included to enable consistent evolution in this important area. The ability to monitor and control the application of behaviors to data objects is a major goal of the XFDU Version 2 Recommended Standard.

10.2 XML SCHEMA FOR BEHAVIOR OBJECTS

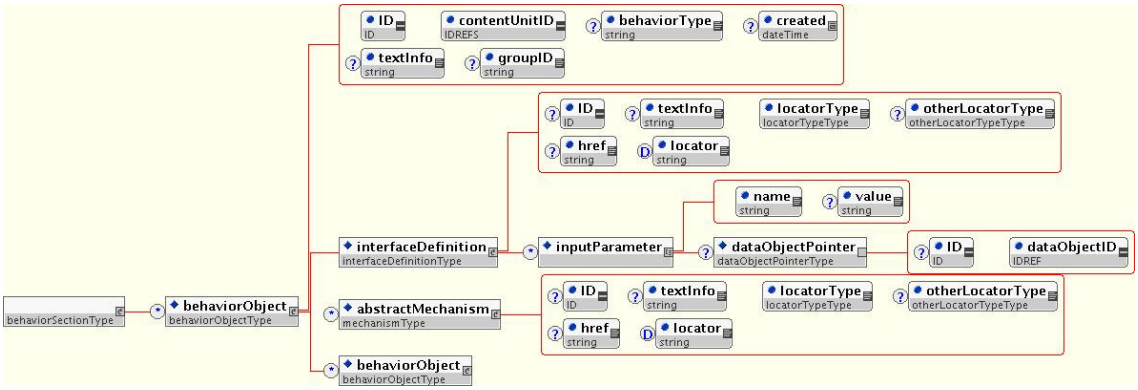


Figure 10-1: behaviorObjectType Schema Diagram

```
<xsd:complexType name="interfaceDefinitionType">
  <xsd:annotation>
    <xsd:documentation>interfaceDefinitionType: interface definition object. The
      interface definition type contains a pointer to an abstract
      definition of a set of related behaviors. These abstract
      behaviors can be associated with the content of a XFDU object.
      The interface definition element will be a pointer to another
      object (an interface definition object). An interface definition
      object could be another XFDU object, or some other entity (e.g.,
      a WSDL source). Ideally, an interface definition object should
      contain metadata that describes a set of behaviors or methods.
      It may also contain files that describe the intended usage of
      the behaviors, and possibly files that represent different
      expressions of the interface definition.
      interfaceDefinition extends from referenceType and adds ability of specifying inputParameter
      that can be either just a string value or pointer to the content in this package
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="xfdu:referenceType">
      <xsd:sequence>
        <xsd:element name="inputParameter" minOccurs="0" maxOccurs="unbounded">
          <xsd:complexType mixed="true">
            <xsd:sequence>
              <xsd:element name="dataObjectPointer" type="xfdu:dataObjectPointerType" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="name" use="required" type="xsd:string"/>
            <xsd:attribute name="value" type="xsd:string"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="behaviorObjectType">
  <xsd:annotation>
    </xsd:documentation>
      behaviorObjectType: Complex Type for Behaviors. A
      behavior object can be used to associate executable behaviors
      with content in the XFDU object. A behavior object has an
        interface definition element that represents an abstract
```

```
        definition of the set of behaviors represented by a particular
      behavior object. An behavior object may have the following
      attributes:
        1. ID: an XML ID for the element
        2. structID: IDREF Enables Behavior to point to Content Units or other Manifest types
        3. behaviorType: a behavior type identifier for a given set of related behaviors.
        4. created: date this behavior object of the XFDU object was created.
        5. textInfo: a description of the type of behaviors this object  represents.
        6. groupID: an identifier that establishes a correspondence between this behavior object and other behavior
      Behavior  object may also include another behavior object for chaining of behaviors.
      Concrete implementation of mechanism have to be used in the instance document.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="interfaceDefinition" type="xfdu:interfaceDefinitionType"/>
    <xsd:element ref="xfdu:abstractMechanism" minOccurs="0" maxOccurs="unbounded"/>
    <xsd:element name="behaviorObject" type="xfdu:behaviorObjectType" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="ID" use="required" type="xsd:ID"/>
  <xsd:attribute name="contentUnitID" use="required" type="xsd:IDREFS"/>
  <xsd:attribute name="behaviorType" type="xsd:string"/>
  <xsd:attribute name="created" type="xsd:dateTime"/>
  <xsd:attribute name="textInfo" type="xsd:string"/>
  <xsd:attribute name="groupID" type="xsd:string"/>
</xsd:complexType>

<xsd:element name="abstractMechanism" type="xfdu:mechanismType" abstract="true">
  <xsd:annotation>
    <xsd:documentation>abstractMechanism is abstract implementation of
      mechanismType. It cannot be instantiated in the instance
      document. Instead, concrete implementations would have to be
      used which are declared part of mechanism substitutionGroup
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>

<xsd:complexType name="mechanismType">
  <xsd:annotation>
    <xsd:documentation>mechanismType: executable mechanism. An element of  mechanismType
      contains a pointer to an executable code module that
      implements a set of behaviors defined by an interface
      definition. The mechanism element will be a pointer to another
      object (a mechanism object). A mechanism object could be another
      XFDU object, or some other entity (e.g., a WSDL source). A
      mechanism object should contain executable code, pointers to
      executable code, or specifications for binding to network
      services (e.g., web services).
      mechanismType is declared as base type for concrete implementations of mechanism
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="xfdu:referenceType"/>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="behaviorSectionType">
  <xsd:sequence>
    <xsd:element name="behaviorObject" type="xfdu:behaviorObjectType" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

10.3 EXAMPLE OF DEFINING AN INTERFACE AND PARAMETER

The following example demonstrates a behavior object that encapsulates behavior for mpeg processing. The interface for the behavior is described via WSDL content pointed to by interfaceDefinition element. InputParameter element specifies an mpeg21 data object as input for this behavior.

```
<behaviorSection>
    <behaviorObject ID="behaviorObject" contentUnitID="cu1">
      <interfaceDefinition locatorType="URL" href="http://sindbad.gsfc.nasa.gov/processmpg21.wsdl">
        <inputParameter name="mpeg21Input">
          <dataObjectPointer dataObjectID="mpeg21"/>
        </inputParameter>
      </interfaceDefinition>
    </behaviorObject>
  </behaviorSection>
```

11 FULL XML SCHEMA –NORMATIVE/RULING

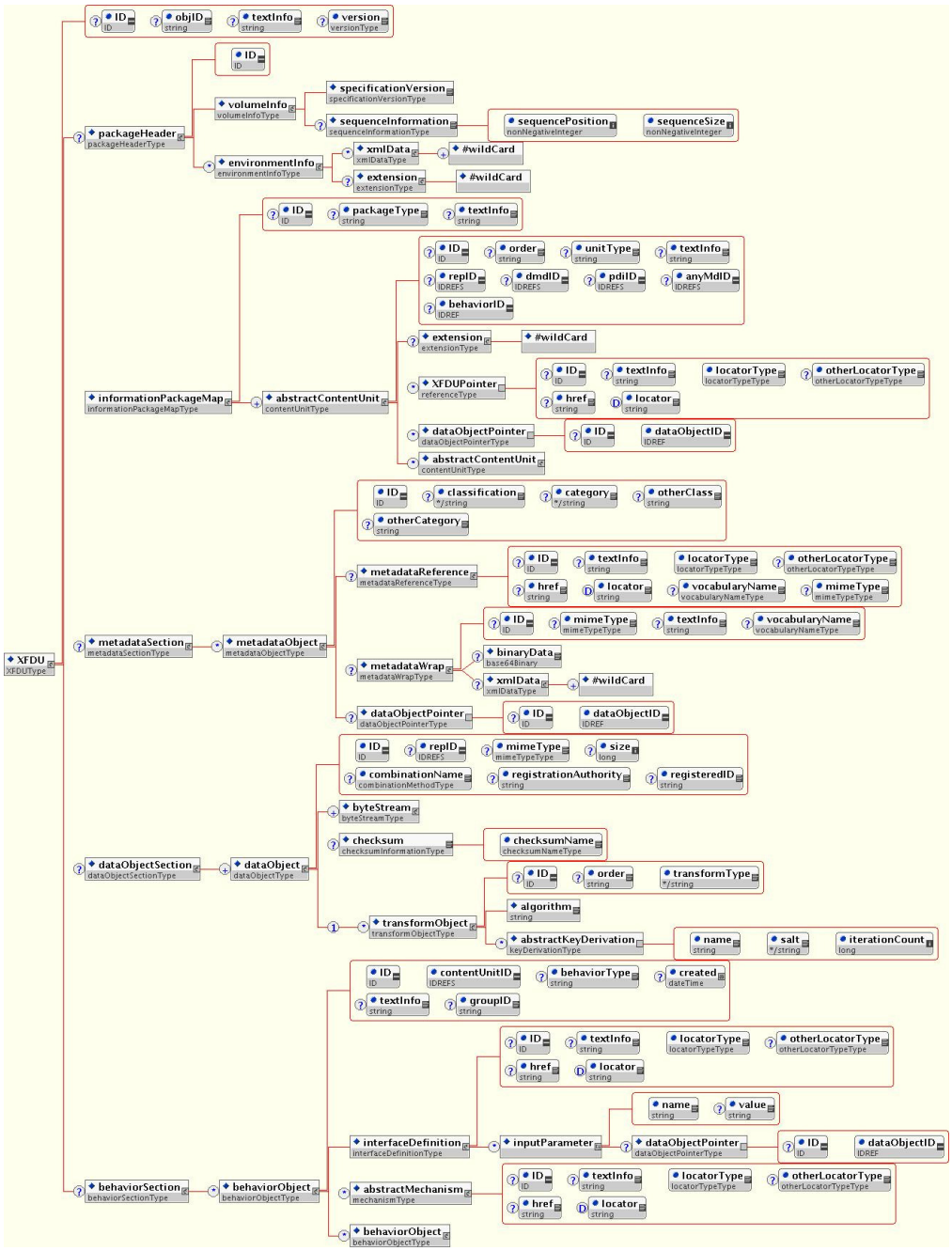


Figure 11-1: Full XFDU Schema Diagram

```
<?xml version="1.0" encoding="UTF-8"?>
<!--Conforms to w3c http://www.w3.org/2001/XMLSchema--><xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xfdu="urn:ccsds:schema:xfdu:1" targetNamespace="urn:ccsds:schema:xfdu:1" elementFormDefault="unqualified"
attributeFormDefault="unqualified">
  <xsd:simpleType name="locatorTypeType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="URL"/>
      <xsd:enumeration value="OTHER"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="otherLocatorTypeType">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
  <xsd:attributeGroup name="LOCATION">
    <xsd:annotation>
      <xsd:documentation>
        This attribute group aggregates attributes that can be used for specifying type of location
        This group includes following attributes:
        locatorType specifies location type (URL or OTHER)
        otherLocatorType specifies location type in case locatorType has value of OTHER
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="locatorType" use="required" type="xfdu:locatorTypeType">
    </xsd:attribute>
    <xsd:attribute name="otherLocatorType" type="xfdu:otherLocatorTypeType"/>
  </xsd:attributeGroup>
  <xsd:attributeGroup name="registrationGroup">
    <xsd:annotation>
      <xsd:documentation>
        This attribute group aggregates attributes that can be used for specifying
        registration information.
        This group includes following attributes:
        registrationAuthority - the authority that issued the registration
        registeredId - the id for the registration
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="registrationAuthority" type="xsd:string"  use="optional"/>
    <xsd:attribute name="registeredID" type="xsd:string" use="optional"/>
  </xsd:attributeGroup>
  <xsd:simpleType name="vocabularyNameType">
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
  <xsd:simpleType name="versionType">
    <xsd:annotation>
      <xsd:documentation>
        Entity of this type is used to indicated version of XFDU XML schema
      </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
  <xsd:simpleType name="mimeTypeType">
    <xsd:restriction base="xsd:string">
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="checksumNameType">
    <xsd:restriction base="xsd:string">
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="combinationMethodType">
    <xsd:restriction base="xsd:string">
```

```
      <xsd:enumeration value="concat"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:attribute name="namespace" type="xsd:string"/>
  <xsd:complexType name="extensionType">
    <xsd:annotation>
      <xsd:documentation>
        allows third parties to define extensions to the XFDU from a namespace
        controlled by the third  party
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:any namespace="##other" processContents="lax"/>
    </xsd:sequence>
    <xsd:anyAttribute namespace="##other" processContents="lax"/>
  </xsd:complexType>

  <xsd:complexType name="sequenceInformationType">
    <xsd:annotation>
      <xsd:documentation>
        An element of this type encapsulates information about the position of the encapsulating XFDU
        package In a sequence of physical XFDU packages that form the identified logical XFDU unit.
        The sequenceInformation element is a string that acts as an identifier for the logical XFDU.
        SequenceInformationType has two mandatory attributes:
        1. sequencePosition - the position of this XFDU package in the sequence; if 0 is specified
        and sequenceSize is unknown, it means that it is last in the sequence
        2. sequenceSize - the total number of packages in the sequence; if its value is 0 this means
        size is unknown
  </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
      <xsd:extension base = "xsd:string">
        <xsd:attribute name = "sequencePosition" type = "xsd:nonNegativeInteger" use="required"/>
        <xsd:attribute name = "sequenceSize" type = "xsd:nonNegativeInteger" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
</xsd:complexType>
  <xsd:complexType name="referenceType">
    <xsd:annotation>
      <xsd:documentation>
        locator attribute allows finer granularity within location specified in href
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="ID" type="xsd:ID"/>
    <xsd:attribute name="textInfo" type="xsd:string"/>
    <xsd:attributeGroup ref="xfdu:LOCATION"/>
    <xsd:attribute name="href" type="xsd:string"/>
    <xsd:attribute name="locator" type="xsd:string" use="optional" default="/"/>
  </xsd:complexType>
  <xsd:complexType name="checksumInformationType">
    <xsd:annotation>
      <xsd:documentation>
      An element of this type would convey checksum information:
        The value of the checksum element is the result of the checksum
        The value of the checksumName attribute is the name of checksum algorithm used to compute the value
      </xsd:documentation>
    </xsd:annotation>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="checksumName" type="xfdu:checksumNameType" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
```

```
  <xsd:complexType name="metadataObjectType">
    <xsd:annotation>
      <xsd:documentation>metadataObjectType Complex Type A generic
      framework for pointing to/including metadata within a XFDU
      document, a la Warwick Framework. An metadataObject element may have the
      following attributes:
      1. ID: an XML ID for this element.
      2. classification - concrete type of metadata represented by this element of metadataObjectType
      3. category - type of metadata class to which this metadata belongs (e.g., DMD.REP, etc.)
      4. otherClass - type of metadata in case classification contains value of "OTHER"
      5. otherCategory - type of metadata class in case category contains value of "OTHER"
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="metadataReference" type="xfdu:metadataReferenceType" minOccurs="0"/>
      <xsd:element name="metadataWrap" type="xfdu:metadataWrapType" minOccurs="0"/>
      <xsd:element name="dataObjectPointer" type="xfdu:dataObjectPointerType" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="ID" use="required" type="xsd:ID"/>
    <xsd:attribute name="classification">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="DED"/>
          <xsd:enumeration value="SYNTAX"/>
          <xsd:enumeration value="FIXITY"/>
          <xsd:enumeration value="PROVENANCE"/>
          <xsd:enumeration value="CONTEXT"/>
          <xsd:enumeration value="REFERENCE"/>
          <xsd:enumeration value="DESCRIPTION"/>
          <xsd:enumeration value="OTHER"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="category">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="REP"/>
          <xsd:enumeration value="PDI"/>
          <xsd:enumeration value="DMD"/>
          <xsd:enumeration value="OTHER"/>
          <xsd:enumeration value="ANY"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="otherClass" type="xsd:string"/>
    <xsd:attribute name="otherCategory" type="xsd:string"/>
  </xsd:complexType>
  <xsd:simpleType name="specificationVersionType">
    <xsd:annotation>
      <xsd:documentation>
        An entity of this type is used to indicated CCSDS-bound version of XFDU specification
      </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
<xsd:complexType name="packageHeaderType">
    <xsd:annotation>
      <xsd:documentation>packageHeaderType: Complex Type for metadata about the
      mapping of the logical packages to the physical structures. The
      package header type has two elements:
-volumeInfo – contains  XFDU volume related metadata (.i.e., XFDU specification version
and sequence information
        - environmentInfo – contains application specific information either defined by an extension of the XFDU
```

```
            Schema or by freeform XML.
          packageHeaderType has a single attribute, ID: an XML ID.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
        <xsd:element name="volumeInfo" type="xfdu:volumeInfoType"/>
        <xsd:element name="environmentInfo" type="xfdu:environmentInfoType" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:attribute name="ID" type="xsd:ID" use="required"/>
    </xsd:complexType>
  <xsd:complexType name="volumeInfoType">
      <xsd:annotation>
        <xsd:documentation>
          Contains XFDU software related system information, including one mandatory element
                - specificationVersion, which specifies the version of the XFDU specification to which this manifest complies.
          Additionally it has one optional element
                -sequenceInformation that holds information about the sequence
          of XFDUs and the position of the current one in it.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
        <xsd:element name="specificationVersion" type="xfdu:specificationVersionType" minOccurs="1" maxOccurs="1"/>
        <xsd:element name="sequenceInformation" type="xfdu:sequenceInformationType" minOccurs="0" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  <xsd:complexType name="environmentInfoType">
      <xsd:annotation>
        <xsd:documentation>
          Environment info provides meta information related to the environment where the XFDU was created.
          Since environment information may be specific to a concrete XFDU producer, environment information can have
          only two optional elements:
          -xmlData - can hold application specific information.as well-formed XML
          - extension element -that serves as extension point for other applications
        </xsd:documentation>
      </xsd:annotation>
        <xsd:sequence>
          <xsd:element name="xmlData" type="xfdu:xmlDataType" minOccurs="0" maxOccurs="unbounded"/>
          <xsd:element name="extension" type="xfdu:extensionType" minOccurs="0"/>

        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="metadataReferenceType">
      <xsd:annotation>
        <xsd:documentation>metadataReferenceType: metadata reference.
        An element of metadataReferenceType is a
        generic element used throughout the XFDU schema to provide a
        pointer to metadata which resides outside the XFDU document.
        metadataReferenceType
        has the following attributes:
        1. ID: an XML ID;
        2. locatorType: the type of locator contained in the body of the element;
        3. otherLocatorType: a string indicating an alternative type of locator when
          the locatorType attribute value is set to "OTHER.";
        4. href: actual location (e.g., URL)
        5. mimeType: the MIME type for the metadata being pointed at;
        6. vocabularyName: the name of the well known metadata standard vocabulary used to being pointed at (e.g., FITS);
        7. textInfo: a label to display to the viewer of the XFDU document identifying the metadata;
        and NB: metadataReference is an empty element. The location of the
        metadata must be recorded in the href attribute.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:complexContent>
        <xsd:extension base="xfdu:referenceType">
```

```
        <xsd:attribute name="vocabularyName" type="xfdu:vocabularyNameType"/>
        <xsd:attribute name="mimeType" type="xfdu:mimeTypeType"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="xmlDataType">
    <xsd:annotation>
      <xsd:documentation>A wrapper to contain arbitrary XML content.</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:any namespace="##any" processContents="lax" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="fileContentType">
    <xsd:annotation>
      <xsd:documentation>
        fileContentType encapsulates and aggregates a type that can have a choice of either
        binary or xml data
      </xsd:documentation>
    </xsd:annotation>
    <xsd:choice>
      <xsd:element name="binaryData" type="xsd:base64Binary" minOccurs="0">
        <xsd:annotation>
          <xsd:documentation>A wrapper to contain Base64 encoded metadata.</xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="xmlData" type="xfdu:xmlDataType" minOccurs="0"/>
    </xsd:choice>
    <xsd:attribute name="ID" type="xsd:ID"/>
  </xsd:complexType>
  <xsd:complexType name="metadataWrapType">
    <xsd:annotation>
      <xsd:documentation>metadataWrapType: metadata wrapper. An element of metadataWrapType is a
      generic element used throughout the XFDU schema to allow the
      encoder to place arbitrary metadata conforming to other
      standards/schema within a XFDU document.  The metadataWrapType
      can have the following attributes:
      1. ID: an XML ID for this element;
      2. mimeType: the MIME type for the metadata contained in the element;
      3. vocabularyName: the type of metadata contained (e.g., MARC, EAD, etc.);
      4. textInfo: a label to display to the viewer of the XFDU document identifying the metadata.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="xfdu:fileContentType">
        <xsd:attribute name="mimeType" type="xfdu:mimeTypeType"/>
        <xsd:attribute name="textInfo" type="xsd:string"/>
        <xsd:attribute name="vocabularyName" type="xfdu:vocabularyNameType"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="dataObjectPointerType">
    <xsd:annotation>
      <xsd:documentation>
        The dataObjectPointerType is a type that can be used to reference dataObjects by dataObjectID.
        The dataObjectPointerType has two attributes:
        1. ID: an XML ID for this element; and
        2. dataObjectID: an IDREF to a dataObject element
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="ID" type="xsd:ID"/>
    <xsd:attribute name="dataObjectID" use="required" type="xsd:IDREF"/>
  </xsd:complexType>
```

```
<xsd:complexType name="keyDerivationType">
  <xsd:annotation>
    <xsd:documentation>keyDerivationType contains the information
    that was used to derive the encryption key for this dataObject.
    Key derivation type contains:
     name - name of algorithm used
    salt - 16-byte random seed used for that algorithm initialization
    iterationCount - number of iterations used by the algorithm to derive the key
    </xsd:documentation>
  </xsd:annotation>
  <xsd:attribute name="name" use="required" type="xsd:string"/>
  <xsd:attribute name="salt" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:length value="16"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="iterationCount" use="required" type="xsd:long"/>
</xsd:complexType>
<xsd:element name="abstractKeyDerivation" type="xfdu:keyDerivationType" abstract="true">
  <xsd:annotation>
    <xsd:documentation>
    abstractKeyDerivation is declared abstract so that it can be used
    for element substitution in cases when the default keyDerivation is not
    sufficient. In order for creating more specific key derivation constructs, one would have to
    extend from keyDerivationType to a concrete type, and then create an element of that new type. Finally,
    in an instance of XML governed by this schema, the reference to keyDerivation in an instance of
    transformObject element would point not to an instance of keyDerivation element, but rather to an instance of the
    custom element. In other words, keyDerivation would be SUBSTITUTED with a concrete key derivation element.
    In cases where default functionality is sufficient, the provided defaultKeyDerivation element can be used for the
    substitution.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="keyDerivation" type="xfdu:keyDerivationType" substitutionGroup="xfdu:abstractKeyDerivation">
  <xsd:annotation>
    <xsd:documentation>
      Default implementation of key derivation type.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:complexType name="transformObjectType">
  <xsd:annotation>
    <xsd:documentation>transformObjectType: transformation information. An element
    of transformObjectType contains all of the information required to reverse the
    transformations applied to the original contents of the dataObject. It
    has two possible subsidiary elements: The algorithm element
    contains information about the algorithm used to encrypt the
    data. The key-derivation element contains the information that
    was used to derive the encryption key for this dataObject It has three
    attributes:
      1. ID: an XML ID
      2. transformType: one of n predefined transformations types.
         Current valid types are compression, encryption, authentication.
      3. order: If there are more than one transformation elements in an dataObject
         this integer indicates the order in which the reversal transformations should be applied.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="algorithm" type="xsd:string">
      <xsd:annotation>
        <xsd:documentation>algorithm element contains information
```

```
            about the algorithm used to encrypt the data.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element ref="xfdu:abstractKeyDerivation" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ID" type="xsd:ID"/>
    <xsd:attribute name="order" type="xsd:string"/>
    <xsd:attribute name="transformType" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="COMPRESSION"/>
          <xsd:enumeration value="AUTHENTICATION"/>
          <xsd:enumeration value="ENCRYPTION"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
  <xsd:complexType name="byteStreamType">
    <xsd:annotation>
      <xsd:documentation>byteStreamType: An element of  byteStreamType
      provides access to the current content of dataObjects for a XFDU
      document. The byteStreamType: has the following four attributes: ID (an XML ID);
      mimeType: the MIME type for the dataObject; size: the size of the dataObject
      in bytes.
      Checksum information provided via optional checksum element.
      The data contained in these attributes is relevant to final state of the data object
      after all possible transformations of the original data.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="fileLocation" type="xfdu:referenceType" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="fileContent" type="xfdu:fileContentType" minOccurs="0"/>
      <xsd:element name="checksum" type="xfdu:checksumInformationType" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="ID" use="optional" type="xsd:ID"/>
    <xsd:attribute name="mimeType" type="xfdu:mimeTypeType"/>
    <xsd:attribute name="size" type="xsd:long"/>
  </xsd:complexType>
  <xsd:complexType name="dataObjectType">
    <xsd:annotation>
      <xsd:documentation>dataObjectType: An element of dataObjectType
      contains current byteStream content and any required data to restore
      them to the form intended for the original designated community.
      It has two possible subsidiary elements: The byteStream element
      provides access to the current content dataObjects for an XFDU
      document. An element of dataObjectType must contain 1 or many byteStream elements
      that may contain a fileLocation element, which provides a pointer to
      a content byteStream, and/or a fileContent element, which wraps an
      encoded version of the dataObject. An element of dataObjectType may contain one or
      more transformation elements that contain all of the
      information required to reverse each transformation applied to
      the dataObject and return the original binary data object.
      The dataObjectType has the following attributes:
      1. ID: an XML ID
      2. mimeType: the MIME type for the dataObject
      3. size: the size of the dataObject in bytes
      4. checksum: a checksum for dataObject. Checksum information provided via optional checksum element.
      5. repID list of representation metadata IDREFs.
        NB: The size, checksum, and mime type are related to the original data before any transformations occurred.
      6. combinationName - specifies how multiple byteStream objects in a single dataObject should be concatenated
      7. registrationGroup attribute group that provides registration information
      </xsd:documentation>
```

```
      </xsd:annotation>
      <xsd:sequence>
        <xsd:element name="byteStream" type="xfdu:byteStreamType" minOccurs="1"
maxOccurs="unbounded"/>
        <xsd:element name="checksum" type="xfdu:checksumInformationType" minOccurs="0" maxOccurs="1"/>
        <xsd:sequence>
          <xsd:element name="transformObject" type="xfdu:transformObjectType" minOccurs="0"
            maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:sequence>
      <xsd:attribute name="ID" type="xsd:ID" use="required"/>
      <xsd:attribute name="repID" type="xsd:IDREFS"/>
      <xsd:attribute name="mimeType" type="xfdu:mimeTypeType"/>
      <xsd:attribute name="size" type="xsd:long"/>
      <xsd:attribute name="combinationName" type="xfdu:combinationMethodType" use="optional"/>
      <xsd:attributeGroup ref="xfdu:registrationGroup"/>
    </xsd:complexType>
    <xsd:complexType name="dataObjectSectionType">
      <xsd:annotation>
        <xsd:documentation>dataObjectSectionType: a container for one or more elements of dataObjectType
        </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
        <xsd:element name="dataObject" type="xfdu:dataObjectType" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="contentUnitType">
      <xsd:annotation>
        <xsd:documentation>ContentUnit Complex Type The XFDU standard
        represents a data package structurally as a series of nested
        content units, that is, as a hierarchy (e.g., a data product,
        which is composed of datasets, which are composed of time
        series, which are composed of records). Every content node in
        the structural map hierarchy may be connected (via subsidiary
        XFDUPointer or dataObjectPointer elements) to information objects which
        represent that unit as a portion of the whole package. The contentUnitType
        is declared as a base type for concrete implementations of contentUnit;
        The content unit element has the following attributes:
        1.ID (an XML ID);
        2.order: a numeric string (e.g., 1.1, 1.2.1, 3,) representation
        of this unit's order among its siblings (e.g., its sequence); order attribute is not meant to be used
        for processing purposes. It is here only for visualization purposes of the potential reader of XML instance.
        It is not guaranteed that any software will take value of order attribute into account. contentUnit nesting is
        the primary means for determining order and level of the content information.
        3.textInfo: a string label to describe this contentUnit to an end
        user viewing the document, as per a table of contents entry
        4.repID: a set of IDREFs to representation information sections
        within this XFDU document applicable to this contentUnit.
        5.dmdID: a set of IDREFS to descriptive information sections
          within this XFDU document applicable to this contentUnit.
        6.pdiID: a set of IDREFS to preservation description information
          sections within this XFDU document applicable to this
          contentUnit
        7.anyMdID: a set of IDREFS to any other metadata sections that do not fit
          rep,dmd or pdi metadata related to this  contentUnit
        8.unitType: a type of content unit (e.g., Application
          Data Unit, Data Description Unit, Software Installation Unit, etc.).
        9. behaviorID-an XML ID reference pointing to associate behavior.
              </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
        <xsd:element name="extension" type="xfdu:extensionType" minOccurs="0"/>
        <xsd:element name="XFDUPointer" type="xfdu:referenceType" minOccurs="0" maxOccurs="unbounded">
```

```
        <xsd:annotation>
          <xsd:documentation>XFDUPointer:XFDU Pointer. The XFDUPointer element allows a
          content unit to be associated with a separate XFDU containing
          the content corresponding with that contentUnit, rather than
          pointing to one or more internal dataObjects. A typical instance of
          this would be the case of a thematic data product that collects
          data products from several instruments observe an event of
          interest. The content units for each instrument datasets might
          point to separate XFDUs, rather than having dataObjects and dataObject
          groups for every dataset encoded in one package. The XFDUPointer
          element may have the following attributes:
          1. ID: an XML ID for this element;
          2. locatorType: the type of location contained in the href attribute;
          3. otherLocatorType: a string to indicate an alternative locator type
            if the locatorType attribute itself has a value of "OTHER."
          NOTE: XFDUPointer is an empty element. The location of the resource pointed to
          MUST be stored in the href attribute.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>
      <xsd:element name="dataObjectPointer" type="xfdu:dataObjectPointerType" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element ref="xfdu:abstractContentUnit" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ID" type="xsd:ID" use="optional"/>
    <xsd:attribute name="order" type="xsd:string"/>
    <xsd:attribute name="unitType" type="xsd:string"/>
    <xsd:attribute name="textInfo" type="xsd:string"/>
    <xsd:attribute name="repID" type="xsd:IDREFS"/>
    <xsd:attribute name="dmdID" type="xsd:IDREFS"/>
    <xsd:attribute name="pdiID" type="xsd:IDREFS"/>
    <xsd:attribute name="anyMdID" type="xsd:IDREFS"/>
    <xsd:attribute name="behaviorID" type="xsd:IDREF"/>

  </xsd:complexType>
  <xsd:element name="abstractContentUnit" type="xfdu:contentUnitType" abstract="true">
    <xsd:annotation>
      <xsd:documentation>abstractContentUnit is abstract implementation of
      contentUnitType. It cannot be instantiated in the instance
      document. Instead, concrete implementations would have to be
      used which are declared part of contentUnit substitutionGroup
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:element name="contentUnit" type="xfdu:contentUnitType"
substitutionGroup="xfdu:abstractContentUnit">
    <xsd:annotation>
      <xsd:documentation>contentUnit is a basic concrete
        implementation of an abstract contentUnit. Its instance can be used
        in the instance document in the place where contentUnit declared
        to be present.
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:complexType name="informationPackageMapType">
    <xsd:annotation>
      <xsd:documentation>informationPackageMapType Complex Type The Information Package Map
        outlines a hierarchical structure for the original object being encoded, using a series of nested contentUnit elements.
      An element of informationPackageMapType has the following attributes:
        1. ID: an XML ID for the element;
        2. packageType: a type for the object. Typical values will be "AIP" for a map which describes
          a complete AIP obeying all constraints and cardinalities in the OAIS reference model.
          "SIP" for a map which describes a Submission Information Package.
        3. textInfo: a string to describe the informationPackageMap to users.
```

```
        4. anyAttribute - wild-carded attribute extension point
           Concrete implementations of abstractContentUnit (contentUnit etc.) must be used in the instance document.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element ref="xfdu:abstractContentUnit" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ID" type="xsd:ID" use="optional"/>
    <xsd:attribute name="packageType" type="xsd:string"/>
    <xsd:attribute name="textInfo" type="xsd:string"/>
    <xsd:anyAttribute namespace="##other" processContents="lax"/>
  </xsd:complexType>
  <xsd:complexType name="interfaceDefinitionType">
    <xsd:annotation>
      <xsd:documentation>interfaceDefinitionType: interface definition object. The
        interface definition type contains a pointer to an abstract
        definition of a set of related behaviors. These abstract
        behaviors can be associated with the content of a XFDU object.
        The interface definition element will be a pointer to another
        object (an interface definition object). An interface definition
        object could be another XFDU object, or some other entity (e.g.,
        a WSDL source). Ideally, an interface definition object should
        contain metadata that describes a set of behaviors or methods.
        It may also contain files that describe the intended usage of
        the behaviors, and possibly files that represent different
        expressions of the interface definition.
        interfaceDefinition extends from referenceType and adds ability of specifying inputParameter
        that can be either just a string value or pointer to the content in this package
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="xfdu:referenceType">
        <xsd:sequence>
          <xsd:element name="inputParameter" minOccurs="0" maxOccurs="unbounded">
            <xsd:complexType mixed="true">
              <xsd:sequence>
                <xsd:element name="dataObjectPointer" type="xfdu:dataObjectPointerType" minOccurs="0"/>
              </xsd:sequence>
              <xsd:attribute name="name" use="required" type="xsd:string"/>
              <xsd:attribute name="value" type="xsd:string"/>
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="behaviorObjectType">
    <xsd:annotation>
      <xsd:documentation>
        behaviorObjectType: Complex Type for Behaviors. A
        behavior object can be used to associate executable behaviors
        with content in the XFDU object. A behavior object has an
        interface definition element that represents an abstract
        definition of the set of behaviors represented by a particular
        behavior object. An behavior object may have the following attributes:
          1. ID: an XML ID for the element
          2. structID: IDREF Enables Behavior to point to Content Units or other Manifest types
          3. behaviorType: a behavior type identifier for a given set of related behaviors.
          4. created: date this behavior object of the XFDU object was created.
          5. textInfo: a description of the type of behaviors this object  represents.
          6. groupID: an identifier that establishes a correspondence between this behavior object and other behavior
        Behavior  object may also include another behavior object for chaining of behaviors.
        Concrete implementation of mechanism have to be used in the instance document.
```

```
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="interfaceDefinition" type="xfdu:interfaceDefinitionType"/>
      <xsd:element ref="xfdu:abstractMechanism" minOccurs="0" maxOccurs="unbounded"/>
      <xsd:element name="behaviorObject" type="xfdu:behaviorObjectType" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="ID" use="required" type="xsd:ID"/>
    <xsd:attribute name="contentUnitID" use="required" type="xsd:IDREFS"/>
    <xsd:attribute name="behaviorType" type="xsd:string"/>
    <xsd:attribute name="created" type="xsd:dateTime"/>
    <xsd:attribute name="textInfo" type="xsd:string"/>
    <xsd:attribute name="groupID" type="xsd:string"/>
  </xsd:complexType>
  <xsd:element name="abstractMechanism" type="xfdu:mechanismType" abstract="true">
    <xsd:annotation>
      <xsd:documentation>abstractMechanism is abstract implementation of
        mechanismType. It cannot be instantiated in the instance
        document. Instead, concrete implementations would have to be
        used which are declared part of mechanism substitutionGroup
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>
  <xsd:complexType name="mechanismType">
    <xsd:annotation>
      <xsd:documentation>mechanismType: executable mechanism. An element of  mechanismType
        contains a pointer to an executable code module that
        implements a set of behaviors defined by an interface
        definition. The mechanism element will be a pointer to another
        object (a mechanism object). A mechanism object could be another
        XFDU object, or some other entity (e.g., a WSDL source). A
        mechanism object should contain executable code, pointers to
        executable code, or specifications for binding to network
        services (e.g., web services).
        mechanismType is declared as base type for concrete implementations of mechanism
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
      <xsd:extension base="xfdu:referenceType"/>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="metadataSectionType">
    <xsd:sequence>
      <xsd:element name="metadataObject" type="xfdu:metadataObjectType" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="behaviorSectionType">
    <xsd:sequence>
      <xsd:element name="behaviorObject" type="xfdu:behaviorObjectType" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="XFDUType">
    <xsd:annotation>
      <xsd:documentation>
      XFDUType Complex Type.
      A XFDU document consists of five possible subsidiary sections:
      packageHeader (XFDU document header), informationPackageMap (content unit section),
      metadataSection (container for metadata objects),
      dataObjectSection (data object section),behaviorSection (behavior section).
      It also has possible attributes:
      1. ID (an XML ID);
      2. objID: a primary identifier assigned to the original source document;
      3. textInfo: a title/text string identifying the document for users;
```

```
      4. version: version to which this XFDU document conforms
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="packageHeader" type="xfdu:packageHeaderType" minOccurs="0"/>
      <xsd:element name="informationPackageMap" type="xfdu:informationPackageMapType"/>
      <xsd:element name="metadataSection" type="xfdu:metadataSectionType" minOccurs="0"/>
      <xsd:element name="dataObjectSection" type="xfdu:dataObjectSectionType" minOccurs="0"/>
      <xsd:element name="behaviorSection" type="xfdu:behaviorSectionType" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="ID" type="xsd:ID"/>
    <xsd:attribute name="objID" type="xsd:string"/>
    <xsd:attribute name="textInfo" type="xsd:string"/>
    <xsd:attribute name="version" type="xfdu:versionType"/>
  </xsd:complexType>
  <xsd:element name="XFDU" type="xfdu:XFDUType"/>
</xsd:schema>
```

ANNEX A

COMPLETE EXAMPLE XFDU

```
<?xml version="1.0" encoding="UTF-8"?>
<xfdu:XFDU xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="urn:ccsds:schema:xfdu:1 http://www.ccsds.org/xfdu/xfdu-1.0.xsd
   cip http://www.ccsds.org/cip/cip-1.0.xsd "
   xmlns:xfdu="urn:ccsds:schema:xfdu:1"
   xmlns:cip="cip">

   <packageHeader ID="packageHeader">
      <volumeInfo>
         <specificationVersion>1.0</specificationVersion>
         <!-- sequence information attribute is specified by producer-->
         <sequenceInformation sequenceSize="10"
sequencePosition="1">producer1.seq10</sequenceInformation>
      </volumeInfo>
      <environmentInfo>
         <xmlData>
            <platform>Linux2.4.22-1.2129.nptl</platform>
         </xmlData>
         <extension>
            <cipIdentificationInformation xmlns="cip"
               producer_ID="NSSDC"
               project_ID="NSSDC_SPMS"
               cip_ID="NSSDC_SPMS-00216_PKG01"
               cip_template_ID="SPMS-00216-DO"
               cip_template_Location="complexTemplateCIP.xml"/>
            </extension>
      </environmentInfo>
   </packageHeader>
 <informationPackageMap ID="informationPackageMap">
      <xfdu:contentUnit ID="cu1" repID = "atdMD" pdiID = "provenance" dmdID = "ECSDMD">
         <dataObjectPointer dataObjectID = "mpeg21"/>
         <xfdu:contentUnit ID="cu2" order = "1" textInfo = "Root content unit for HDF data">
           <xfdu:contentUnit ID="cu3" order = "1.1" pdiID = "provenance" textInfo = "content unit for hdfFile0" dmdID = "ECSDMD">
              <dataObjectPointer dataObjectID = "hdfFile0"/>
           </xfdu:contentUnit>
           <xfdu:contentUnit ID="cu4" order = "1.2" pdiID = "provenance" textInfo = "content unit for hdfFile1" dmdID = "ECSDMD">
              <dataObjectPointer dataObjectID = "hdfFile1"/>
           </xfdu:contentUnit>
           <xfdu:contentUnit ID="cu5" order = "1.3" pdiID = "provenance" textInfo = "content unit for hdfFile2" dmdID = "ECSDMD">
              <dataObjectPointer dataObjectID = "hdfFile2"/>
           </xfdu:contentUnit>
         </xfdu:contentUnit>
         <xfdu:contentUnit ID="cu6" textInfo = "content unit for orbit data">
            <dataObjectPointer dataObjectID = "orbitalData"/>
         </xfdu:contentUnit>
      </xfdu:contentUnit>
      <xfdu:contentUnit ID="cu7" textInfo = "content unit ATD metadata">
         <dataObjectPointer dataObjectID = "ATDMD"/>
      </xfdu:contentUnit>
   </informationPackageMap>
   <metadataSection>
```

```
      <metadataObject ID = "ECSDMD" classification = "OTHER" category = "DMD">
         <metadataReference vocabularyName="OTHER" mimeType = "text/xml" textInfo = "spacecraft description" locatorType = "URL"
href = "file:packagesamples/scenario1/ecsdmd.xml"/>
      </metadataObject>
      <metadataObject ID = "provenance" classification = "PROVENANCE" category = "PDI">
         <metadataReference vocabularyName = "OTHER" mimeType = "text/xml" textInfo = "processing  history  XML file"
            locatorType= "URL" href = "file:packagesamples/scenario1/pdi.xml" />
      </metadataObject>
      <metadataObject ID = "atdMD" classification = "OTHER" category = "REP">
         <dataObjectPointer dataObjectID = "ATDMD"/>
      </metadataObject>
      <metadataObject ID = "mathMLAlgRepMD" classification = "OTHER" category = "REP">
         <metadataWrap vocabularyName = "OTHER" textInfo = "mathML encoding of the algorithm">
            <xmlData>
               <math>
                  <mrow>
                     <mo>det</mo>
                     <mo symmetric = "false" rspace = "0" lspace = "0">|</mo>
                     <mfrac linethickness = "0">
                        <mi>a</mi>
                        <mi>c</mi>
                     </mfrac>
                     <mfrac linethickness = "0">
                        <mi>b</mi>
                        <mi>d</mi>
                     </mfrac>
                     <mo symmetric = "false" rspace = "0" lspace = "0">|</mo>
                     <mo>=</mo>
                     <mi>a</mi>
                     <mi>d</mi>
                     <mo>-</mo>
                     <mi>b</mi>
                     <mi>c</mi>
                     <mo>,</mo>
                  </mrow>
               </math>
            </xmlData>
         </metadataWrap>
      </metadataObject>
   </metadataSection>
   <dataObjectSection>
      <dataObject repID = "mathMLAlgRepMD" ID = "mpeg21">
         <byteStream mimeType = "video/mpeg" ID = "mpeg21AnimData" size = "414131">
            <fileLocation locatorType = "URL" href = "file:packagesamples/scenario1/mpeg21.mpg"/>
            <checksum checksumName="CRC32">b3eb4b34</checksum>
         </byteStream>
      </dataObject>
      <dataObject size = "151672" mimeType = "application/pdf" ID = "ATDMD">
         <byteStream mimeType = "application/octetstream" ID = "atdMDbs" size = "110874">
            <fileLocation locatorType = "URL" href = "file:packagesamples/scenario1/atd.pdf" />
            <checksum checksumName="CRC32">ad78ad5d</checksum>
         </byteStream>
         <checksum checksumName="CRC32">6d0e30ea</checksum>
         <transformObject transformType = "ENCRYPTION">
            <algorithm>blowfish</algorithm>
         </transformObject>
      </dataObject>
      <dataObject mimeType = "application/octetstream" ID = "orbitalData">
         <byteStream ID = "orbitData">
            <fileLocation locatorType = "URL" href = "http://coin.gsfc.nasa.gov:8080/ims-bin/3.0.1/nph-
ims.cgi?msubmit=yes&amp;lastmode=SRCHFORM"/>
```

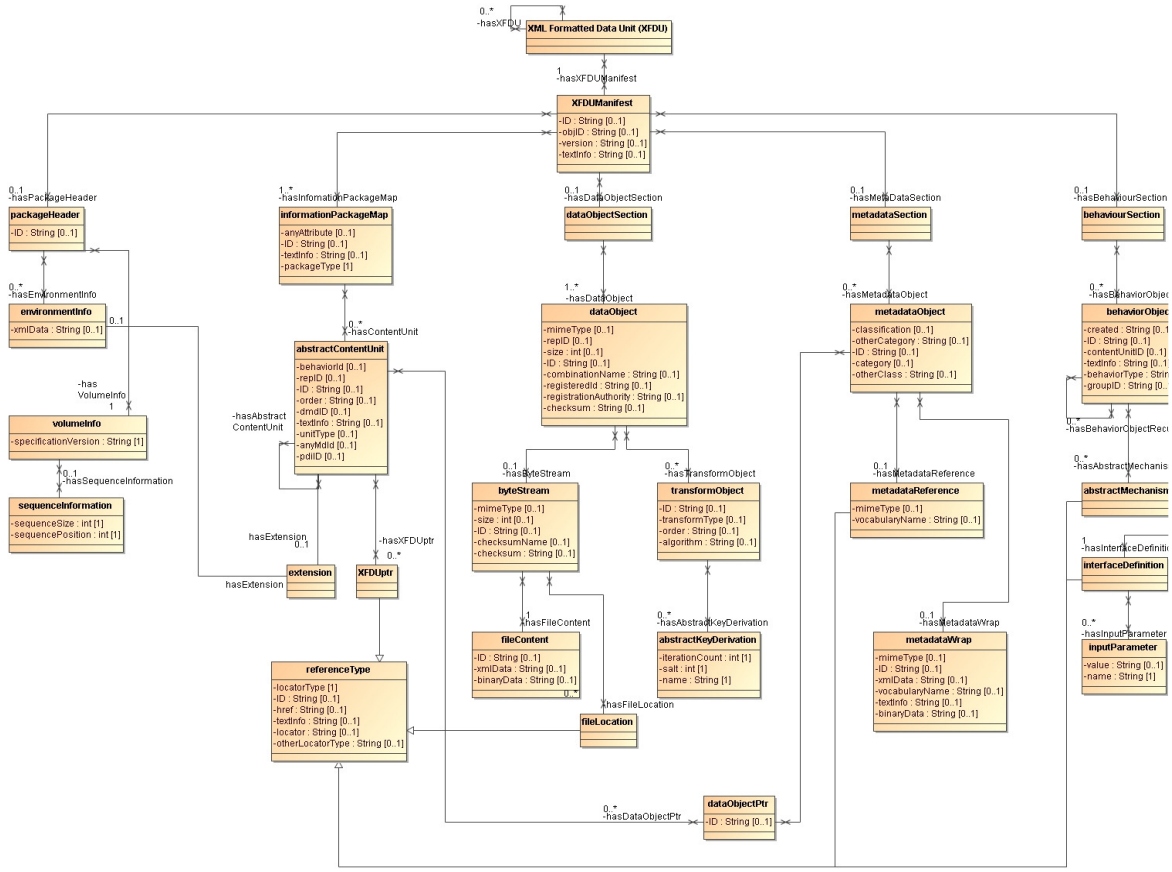
```
            <fileContent>

<binaryData>UEsDBBQACAAIAKqMBC8AAAAAAAAAAAAAAAPAAAAeGZkdS8uY2xhc3NwYXRotZXfS8MwEMff/StK35OugqCwH4h
O0I...</binaryData>
            </fileContent>
            <checksum checksumName="CRC32">b3eb4b34</checksum>
          </byteStream>
        </dataObject>
        <dataObject repID = "atdMD" ID = "hdfFile0">
          <byteStream mimeType = "application/x-hdf" ID = "hdfFile0bs" size = "10455471">
            <fileLocation locatorType = "URL" href = "file:packagesamples/scenario1/mod1.hdf"/>
            <checksum checksumName="CRC32">acab6535</checksum>
          </byteStream>
        </dataObject>
        <dataObject repID = "atdMD" ID = "hdfFile1">
          <byteStream mimeType = "application/x-hdf" ID = "hdfFile1bs" size = "10455471">
            <fileLocation locatorType = "URL" href = "file:packagesamples/scenario1/mod2.hdf"/>
            <checksum checksumName="CRC32">acab6535</checksum>
          </byteStream>
        </dataObject>
        <dataObject repID = "atdMD" ID = "hdfFile2">
          <byteStream mimeType = "application/x-hdf" ID = "hdfFile2bs" size = "10455471">
            <fileLocation locatorType = "URL" href = "file:packagesamples/scenario1/mod3.hdf"/>
            <checksum checksumName="CRC32">acab6535</checksum>
          </byteStream>
        </dataObject>
      </dataObjectSection>
      <behaviorSection>
        <behaviorObject ID="behaviorObject" contentUnitID="cu1">
          <interfaceDefinition locatorType="URL" href="http://sindbad.gsfc.nasa.gov/processmpg21.wsdl">
            <inputParameter name="mpeg21Input">
              <dataObjectPointer dataObjectID="mpeg21"/>
            </inputParameter>
          </interfaceDefinition>
        </behaviorObject>
      </behaviorSection>
    </xfdu:XFDU>
```

ANNEX B

UML FOR XFDU

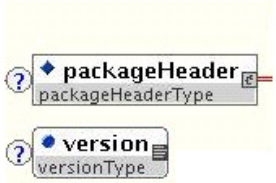
The following UML Diagram is not Normative. It is included to help in understanding the XFDU schema.



ANNEX C

LEGEND FOR XML AUTHORITY FIGURES

Main Graphical Elements



Element, includes the element name in top portion of box and the element type in bottom portion. (Note square corners and diamond in upper left corner of rectangle.)

Attribute, includes the attribute name in top portion of box and the attribute type in bottom portion. (Note square corners and circle in upper left corner of rectangle.)

Type indicators in right of element/attribute

indicates ID or IDREF type

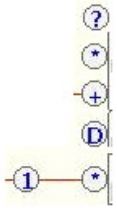
indicates String type (or enumeration of URI)

indicates Numeric type (integer or long)

indicates a Complex Type (i.e., a type made up of other elements)

indicates an empty element (only attributes included with this element)

Occurrence indicator left of element/attribute



Indicates 0 or 1 occurrence (Optional)

Indicates 0 or more occurrences (Optional, but multivalued)

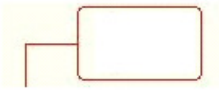
Indicates 1 or more occurrences (Multi-valued mandatory)

Indicates a Default value (may also see an F indicating a Fixed value)

Indicates a 1 to many relationship between elements

If none of these appear before the element or attribute, the indication is that it appears exactly once (Mandatory)

Line connectors



Indicates a set of attributes (rectangles for each of the attributes appears inside the connected box.

Indicates a sequence, all elements between the rectangle connected to the top line and the rectangle connected to the bottom line are included in the sequence. A single line is used if there is only a single child.

Indicates a choice

ANNEX D

INFORMATIVE REFERENCES

[D1] *Standard Formatted Data Units — Structure and Construction Rules*. Recommendation for Space Data System Standards, CCSDS 620.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, May 1992.

[D2] *ASCII Encoded English (CCSD0002)*. Recommendation for Space Data System Standards, CCSDS 643.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, November 1992.

[D3] *Standard Formatted Data Units — Control Authority Procedures*. Recommendation for Space Data System Standards, CCSDS 630.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, June 1993.

[D4] *Standard Formatted Data Units — Control Authority Data Structures*. Recommendation for Space Data System Standards, CCSDS 632.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, November 1994.

[D5] *Standard Formatted Data Units—Referencing Environment*. Recommendation for Space Data System Standards, CCSDS 622.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 1997.

[D6] *Parameter Value Language Specification (CCSD0006 and CCSD0008)*. Recommendation for Space Data System Standards, CCSDS 641.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, June 2000.

[D7] *Reference Model for an Open Archival Information System (OAIS)*. Recommendation for Space Data System Standards, CCSDS 650.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, January 2002.

[D8] Steve DeRose, Eve Maler, and David Orchard, eds. *XML Linking Language (XLink) Version 1.0*. W3C Recommendation. N.p.: W3C, June 2001. <http://www.w3.org/TR/2001/REC-xlink-20010627/>

[D9] "METS—Metadata Encoding and Transmission Standard: Official Web Site." December 11, 2006. *The Library of Congress*. Network Development and MARC Standards Office of the Library of Congress. <http://www.loc.gov/standards/mets/> (12/18/2006)

NOTE – Normative references are listed in 1.7.