Consultative Committee for Space Data Systems

RECOMMENDATION FOR SPACE DATA SYSTEM STANDARDS

XML Formatted Data Unit (XFDU) Structure and Construction Rules

CCSDS [number]

WHITE BOOK

May 15, 2006

AUTHORITY

Issue:
Date:
Location:

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 Recommendations is detailed in *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

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of member space Agencies. 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 **Recommendations** and are not considered binding on any Agency.

This **Recommendation** is issued by, and represents the consensus of, the CCSDS Plenary body. Agency endorsement of this **Recommendation** is entirely voluntary. Endorsement, however, indicates the following understandings:

– Whenever an Agency establishes a CCSDS-related **standard**, this **standard** will be in accord with the relevant **Recommendation**. Establishing such a **standard** does not preclude other provisions which an Agency may develop.

– Whenever an Agency establishes a CCSDS-related standard, the Agency will provide other CCSDS member Agencies with the following information:

  • The **standard** itself.

  • The anticipated date of initial operational capability.

  • The anticipated duration of operational service.

– Specific service arrangements are made via memoranda of agreement. Neither this Recommendation nor any ensuing standard is a substitute for a memorandum of agreement.

No later than five years from its date of issuance, this **Recommendation** 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 **Recommendation** is issued, existing CCSDS-related Agency standards and implementations are not negated or deemed to be non-CCSDS compatible. It is the responsibility of each Agency to determine when such standards or implementations are to be modified. Each Agency is, however, strongly encouraged to direct planning for its new standards and implementations towards the later version of the Recommendation.

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 meets current CCSDS agency requirements, 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 Recommendation 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.
- Instituto Nacional de Pesquisas Espaciais (INPE)/Brazil.
- Japan Aerospace Exploration Agency (JAXA)/Japan.
- National Aeronautics and Space Administration (NASA)/USA.
- Russian Space Agency (RSA)/Russian Federation.

Observer Agencies

- Austrian Space Agency (ASA)/Austria.
- 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.
- Communications Research Laboratory (CRL)/Japan.
- Danish Space Research Institute (DSRI)/Denmark.
- European Organization for the Exploitation of Meteorological Satellites (EUMETSAT)/Europe.
- European Telecommunications Satellite Organization (EUTELSAT)/Europe.
- Federal Science Policy Office (FSPO)/Belgium.
- Hellenic National Space Committee (HNSC)/Greece.
- Indian Space Research Organization (ISRO)/India.
- Institute of Space and Astronautical Science (ISAS)/Japan.
- Institute of Space Research (IKI)/Russian Federation.
- KFKI Research Institute for Particle & Nuclear Physics (KFKI)/Hungary.
- MIKOMTEK: CSIR (CSIR)/Republic of South Africa.
- Korea Aerospace Research Institute (KARI)/Korea.
- Ministry of Communications (MOC)/Israel.
- National Oceanic & Atmospheric Administration (NOAA)/USA.
- National Space Program Office (NSPO)/Taipei.
- Space and Upper Atmosphere Research Commission (SUPARCO)/Pakistan.
- Swedish Space Corporation (SSC)/Sweden.
- United States Geological Survey (USGS)/USA.

DOCUMENT CONTROL

Document	Title and Issue	Date	Status
CCSDS-6xx.x-R-1	XML Formatted Data Unit (XFDU) Structure and Construction Rules	May 2006	

CONTENTS

Section Page

Table of Figure

Figure Page

Table of Tables

Table Page

1 INTRODUCTION

1.1 PURPOSE AND SCOPE

The main purpose of this document is to define a CCSDS Recommendation 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 recommendations (References [1], [2], [3], [4], [5], [6]) to accommodate the current computing environment and meet evolving requirements, and that can be implemented to demonstrate practical, near-term results.

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 "Packaging Techniques" is 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 (Informative)
  - N.4 –Semantics (Semantics that cannot be expressed by the W3C XML Schema Language

There are several notation issues to be understood:

1. 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 Chapter 11

2. 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 XML Schema Language

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

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-B

- 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

1.4 DEFINITIONS

1.4.1 ACRONYMS AND ABBREVIATIONS

AIC	Archival Information Collection
AIP	Archival Information Package
AIU	Archival Information Unit
ASCII	American Standard Code for Information Interchange
CCSDS	Consultative Committee for Space Data Systems
CD-ROM	Compact Disk - Read Only Memory
CORBA	Common Object Request Broker Architecture
CRC	Cyclical Redundancy Check
DIME	Direct Internet Message Encapsulation
DIP	Dissemination Information Package
FITS	Flexible Image Transfer System
GIF	Graphics Interchange Format
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
PDS	Planetary Data System
RDF	Resource Description Format
RPC	Remote Procedure Call
SFDU	Standard Formatted Data Unit
SIP	Submission Information Package
SOAP	Simple Object Access Protocol
UML	Unified Modeling Language
UNICODE	Universal Code
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

1.4.2 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.

Application Data Unit: A Content Unit type where all the objects are related to one or more primary content objects of interest

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: Contains zero or more behavior objects

CCSDS Control Authority: An organization under the auspices of the CCSDS that supports the transfer and usage or SFDUs by providing operational services of registration, archiving, and dissemination of data descriptions. It is comprised of:

- The CCSDS Secretariat supported by the Control Authority Agent
- Member Agency Control Authority Offices

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

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

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.

Exchange Data Unit: A Content Unit type where the objects have been packaged for a reason. It is assumed that all the objects are related though the exact relationship is not one of a set of predefined types.

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 relationshipss 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 Section 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: Refers to 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 aadministrative metadata for the whole XFDU, such as version, operating system, hardware, author, etc, and metadata about transformations and behaviors that must be understood, in particular, transformations which must be reversed to access the data content.

Package Interchange File: A collection of files that have been bundled together into a single container.

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 defintion 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 contain 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: 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.

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.

Xlink: W3C Recommendation that defines XML-conforming syntax for expressing links among XML documents and other Internet resources, and defines some of the behavior of applications that support it.

XML Formatted Data Unit: 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

1.5 CONFORMANCE --- ****LOU TO DO

An ***XFDU Manifest*** is a Manifest that is conformant to the XML Schema specified in this Recommendation (see Section11).

An ***XFDU Package*** is a Package Interchange File that contains an XFDU Manifest and is conformant to the semantics specified in this document

1.6 REFERENCES

1.6.1 NORMATIVE REFERENCES

[XML] eXtensible Markup Language (XML) 1.0 (Second Edition) - W3C Recommendation - October 6, 2000 - Version 1.0 - Copyright © 2000 World Wide Web Consortium (W3C) –

[Xlink] XML Linking Language (XLink) Version 1.0 - W3C Recommendation – June 27, 2001 - Version 1.0 - Copyright © 2001 World Wide Web Consortium (W3C) –

[XML-SCHEMA-STRUCT] XML Schema : Structures - W3C Recommendation - May 2, 2001 - Version 1.0 - Copyright © 2001 World Wide Web Consortium (W3C) –

[XML-SCHEMA-TYPES] XML Schema : Data Types - W3C Recommendation - May 2, 2001 - Version 1.0 - Copyright © 2001 World Wide Web Consortium (W3C) -

XML Authority http://www.extensibility.com/software/metadata/turboxml.jsp

1.6.2 INFORMATIVE REFERENCES

[1] *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. (Also ISO 12175:1994)

[2] *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. (Also ISO 14962:1997)

[3] *Standard Formatted Data Units — Control Authority Procedures*. Recommendation for Space Data Systems Standards, CCSDS 630.0-B-1. Blue Book. Issue1. Washington, D.C., CCSDS, June 1993. (Also ISO 13764:1996)

[4] 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. (Also ISO 15395:1998)

[5] *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. (Also ISO 15888:2000)

[6] *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. (Also ISO 14961:2002)

[7] *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. (Also ISO 14721:2003)

[8] Metadata Encoding and Transmission Standard (METS) <http://www.loc.gov/standards/mets/>

2 OVERVIEW OF PROPOSED 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 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.

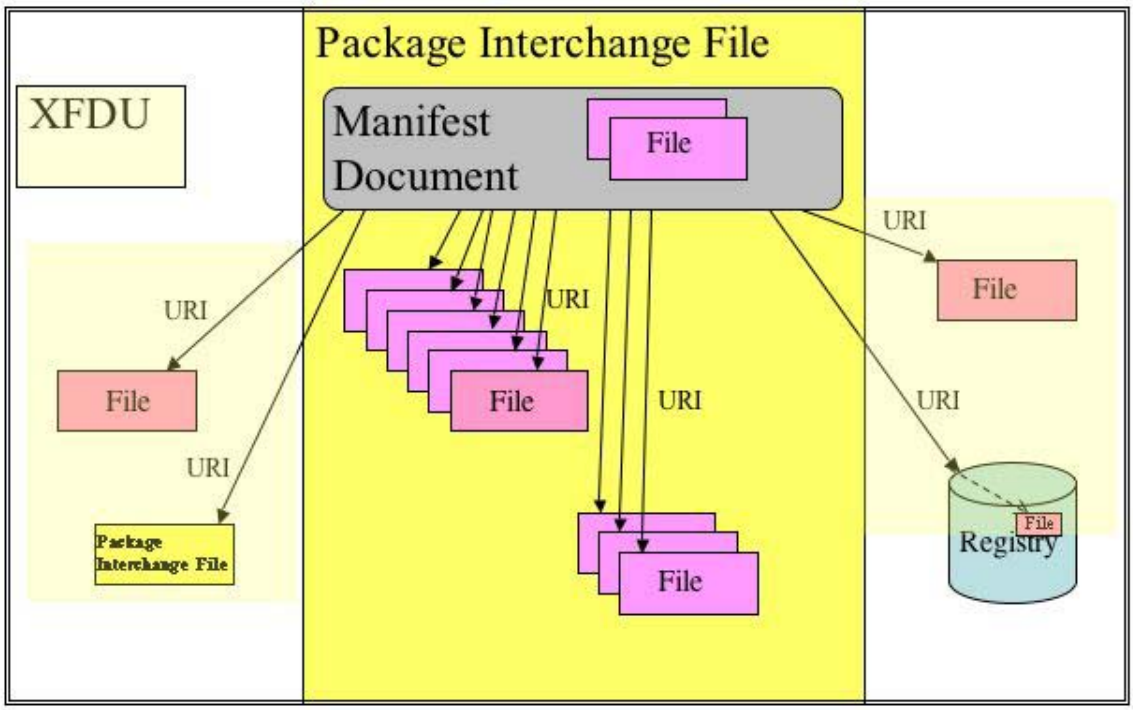


Figure 1 Environment/Conceptual View of an XFDU

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 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 Section11).

An ***XFDU Package*** is a Package Interchange File that contains an XFDU Manifest and is conformant to the semantics specified in this document Figure 2 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 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 within the package or outside, accessed. 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 les 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.

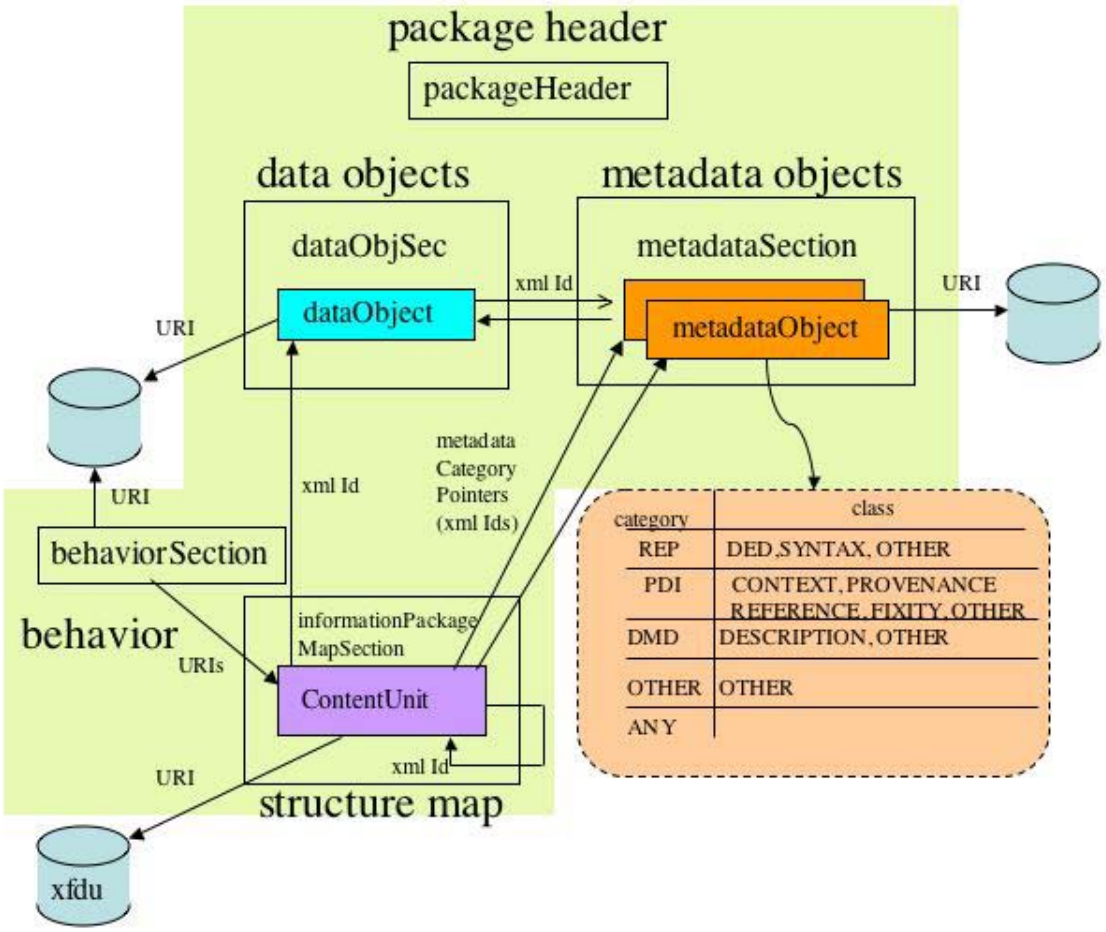


Figure 2: XFDU Manifest Logical View

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.

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. Abststract element for mechanisms and substitution group for 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

Table 1:XFDU Functionality by Recommendation Issue

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 [REF14] XML schema diagram of the XFDU is shown as Figure 3. .

1. **Package Header** (packageHeader): Administrative metadata for the whole XFDU Package, such as version, operating system, hardware, author, etc, and metadata about transformations and behaviors that must be understood; in particular, transformations which must be reversed to access the data content.
2. **Information Package Map (informationPackageMap)** 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
3. **Data Object Section (dataObjectSection)** contains a number of dataObject elements. A Data Object logically contains a byte stream and 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.
4. **Metadata Section (**metadataSection): This section 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.
5. **Behavior Section (behaviorSection)** 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 3: 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: an 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 classes are reused throughout the XFDU schema to represent some recurring structures:

- referenceType - Entity that can reference a resource via a URI.
- dataObjectPointer - An empty element that references dataObjects using the XMLID.
- fileContentType - Entity that encapsulates either binary or XML arbitrary content.
    - binaryData - Entity that encapsulates base64 encoded data (e.g. binary data).
    - xmlData - Entity that encapsulates 1 to many pieces of arbitrary XML data.
- checksumInformationType - Entity that identifies the checksum type.

As discussed in Section 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

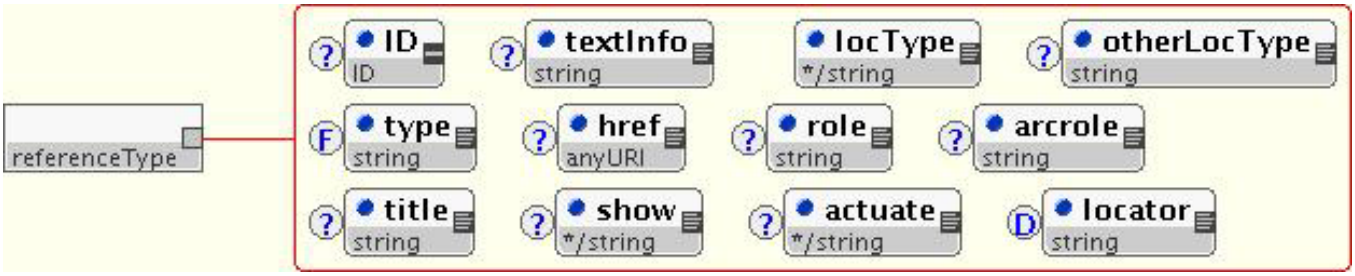


Figure 4 referenceType Schema Diagram

```
<xsd:complexType name="referenceType">
    <xsd:attribute name="ID" type="xsd:ID"/>
    <xsd:attribute name="textInfo" type="xsd:string"/>
    <xsd:attributeGroup ref="LOCATION"/>
    <xsd:attributeGroup ref="xlink:simpleLink"/>
    <xsd:attribute name="locator" type="xsd:string" use="optional" default="/"/>
</xsd:complexType>
```

Figure 5: dataObjectPointerTypeSchema Diagram

```
<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>>
```

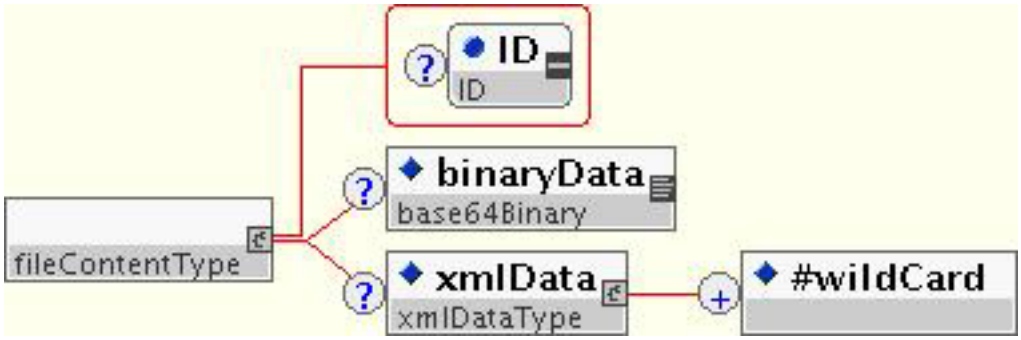


Figure 6 fileContentType/binaryData/xmlData[1]Schema Diagram

```
<xsd:complexType xmlns:xsd="http://www.w3.org/2001/XMLSchema"  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
```

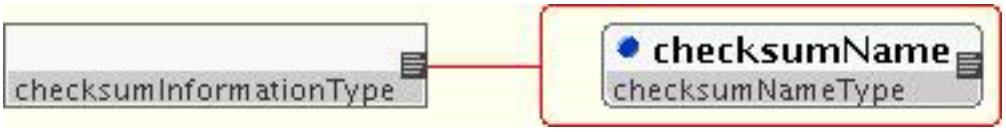


Figure 7 checksumInformationType

```
 <xsd:complexType xmlns:xsd="http://www.w3.org/2001/XMLSchema" 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 he name of checksum algorithm used to compute the value
      </xsd:documentation>
```

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

```
  </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>
```

5 PACKAGE HEADER TYPE

5.1 OVERVIEW

The Package Header is an XML Complex Type for administrative and technical requirements metadata that apply to the containing XFDU package. The package header section consists of three optional sections:

- Environment Information (environmentInfo): specification of the hardware and software platform which created this package and other administrative data such as creator and modification authority which applies to the whole instance of the XFDU

- Behavior Information (behaviorInfo): specification of the behavior mechanism related metadata including the mechanism type, general description, namespace and any other metadata required to specify the behavior.

- Transformation Information (transformInfo): specification of the names, classifications, parameter names/types and any other information needed to reverse transformations used in the XFDU.

5.2 XML SCHEMA packageHeader Type

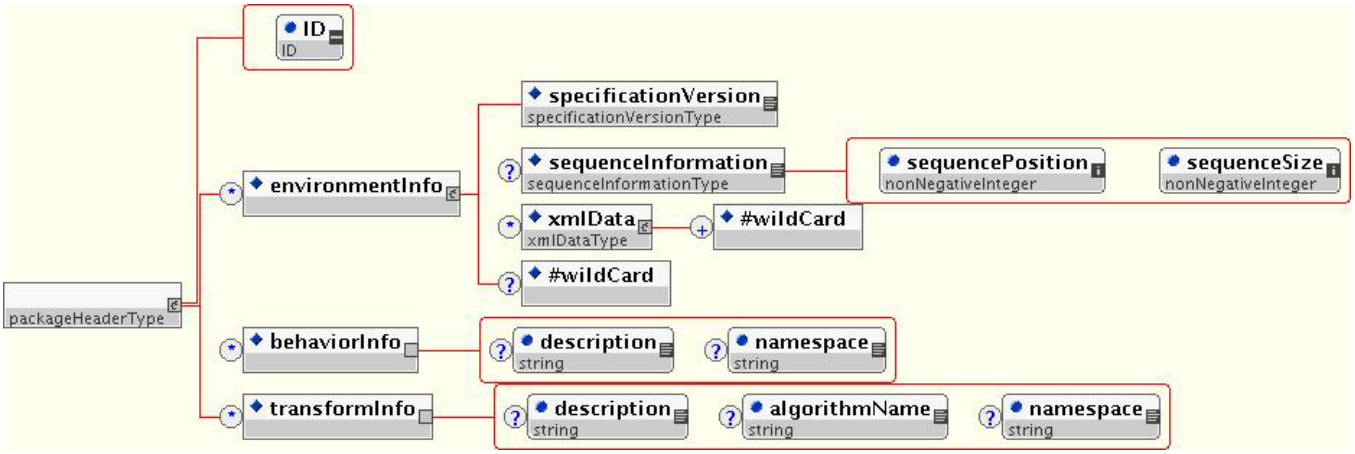


Figure 8: packageHeaderType Schema Diagram

```
<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 section consists of three possible subsidiary sections:
            environmentInfo -specification of technical and administrative metadata that are valid for the entire XFDU (e.g., the
            hardware and software platform which created this package),
```

```
        behaviorInfo – a description of behaviors identified in the containing XFDU and
        transformInfo (the names, classifications, parameter names/types and any other information
        needed to reverse  transformations used in the containing XFDU).
        packageHeaderType has a single attribute, ID: an XML ID.
    </xsd:documentation>
  </xsd:annotation>
  <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 some logical XFDU unit.
      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:simpleType name="specificationVersionType">
  <xsd:annotation>
    <xsd:documentation>
      Entity of this type is used to indicated CCSDS document issue of XFDU specification
    </xsd:documentation>
  </xsd:annotation>
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
  <xsd:sequence>
    <xsd:element name="environmentInfo" minOccurs="0" maxOccurs="unbounded">
      <xsd:annotation>
        <xsd:documentation>environmentInfo: The environmentInfoprovides a wrapper around a generic metadata
              section that should contain should contain technical and administrative metadata that is valid for the
              entire XFDU instance. The environmentInfo contains two defined elements:
                  1. specificationVersion,- which specifies CCSDS Document Number that served as the normative
                  specification for the containing XFDU
                  2. sequenceInformation An element of this type encapsulates information about the position of the
                  encapsulating XFDU packageIn a sequece of physical XFDU packages that form
                  a  logical XFDU package
        </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
        <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:element name="xmlData" type="xfdu:xmlDataType" minOccurs="0" maxOccurs="unbounded"/>
          <xsd:any namespace="##other" minOccurs="0" processContents="strict"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="behaviorInfo" minOccurs="0" maxOccurs="unbounded">
      <xsd:annotation>
        <xsd:documentation>
          behaviorInfo contains:
          description - general description
          namespace - namespace of the specified technology
        </xsd:documentation>
```

```
      </xsd:annotation>
      <xsd:complexType>
        <xsd:attribute name="description" type="xsd:string"/>
        <xsd:attribute ref="xfdu:namespace"/>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="transformInfo" minOccurs="0" maxOccurs="unbounded">
      <xsd:annotation>
        <xsd:documentation>transformInfo (the names, classifications, parameter
        names/types and any other information needed to reverse
        transformations used in the XFDU)
        transformInfo contains:
        description - general description
        algorithmName -name of transformation algorithm
        namespace - namespace of the specified technology
      </xsd:annotation>
      <xsd:complexType>
        <xsd:attribute name="description" type="xsd:string"/>
        <xsd:attribute name="algorithmName" type="xsd:string"/>
        <xsd:attribute ref="xfdu:namespace"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="ID" type="xsd:ID" use="required"/>
</xsd:complexType>
```

5.3 EXAMPLES

5.3.1 PACKAGE HEADER USING sequenceInformation

```
<packageHeader ID="packageHeader">
   <environmentInfo>
      <specificationVersion>
         <xmlData>
            <version>1.0</version>
         </xmlData>
      </specificationVersion>
      <!-- sequence information attribute is specified by producer-->
      <sequenceInformation sequenceSize="10" sequencePosition="1">producer1.seq10</sequenceInformation>
      <xmlData>
         <platform>Linux2.4.22-1.2129.nptl</platform>
      </xmlData>
   </environmentInfo>
   <transformInfo algorithmName = "blowfish" description = "encryption"/>
</packageHeader>
```

5.4 SEMANTICS

1. Currently environmentInfo is a list of XML element and attributes specified by the XFDU producer. In the final version of this Recommendation this may become a controlled vocabulary.

6 CONTENT UNIT

6.1 OVERVIEW

A ***Content Unit*** (contentUnit) 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

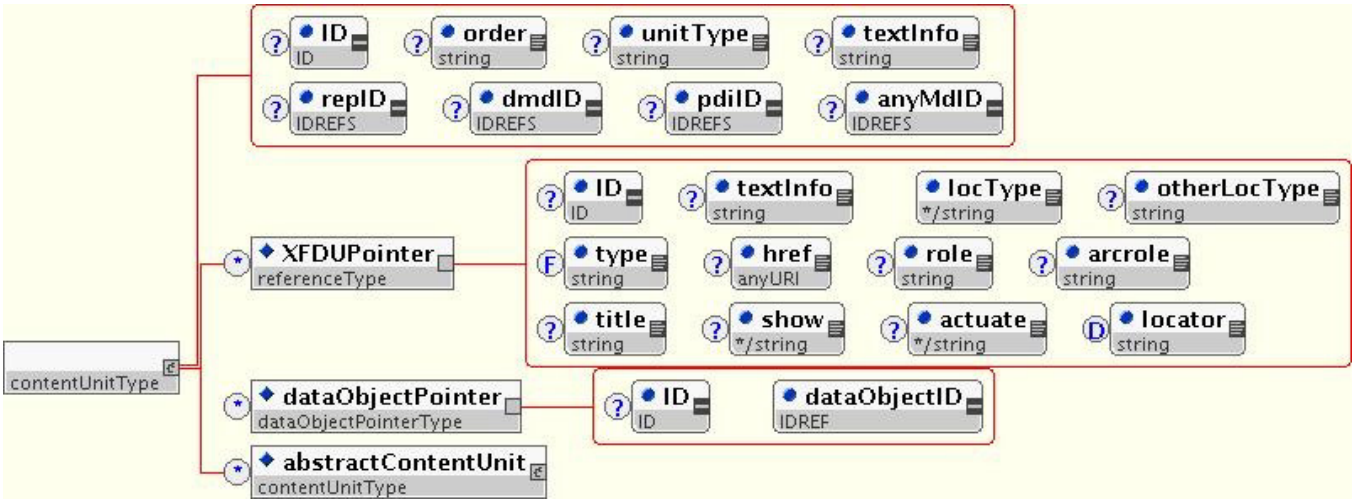


Figure 9: contentUnitType XML Schema

```
<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 content
      units element has the following attributes:
      1.ID (an XML ID);
      2.order: an numeric string (e.g., 1.1, 1.2.1, 3,) annotation of this unit's order among its siblings The order
      attribute is not meant to be used for processing purposes. It is only a visualization aid for 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 metdata related to this  contentUnit
    8.unitType: a type of content unit (e.g., Application
    Data Unit, Data Description Unit, Software Installation Unit, etc.).
    contentUnitType is declared as a base type for concrete implementations of contentUnit.
   </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <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. locType: the type of locater contained in the xlink:href attribute;
            3. otherLocType: a string to indicate analternative locType if the locType attribute itself has a value  of "OTHER.
            4. "xlink:href: see XLink standard (http://www.w3.org/TR/xlink)
            5. xlink:role: ""
            6. ” xlink:arcrole: ""
            7. xlink:title: "
            8. " xlink:show: "" xlink:actuate:
            "" NOTE: XFDUPointeris an empty element. The location of the resource pointed to MUST be stored in the
            xlink:href element.
        </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:complexType>
```

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 SUBMISSION INFORMATION PACKAGE CONTENT UNIT

The following

```
<!--SIP content unit is an extension of contentUnitType and
   part of abstractContentUnit substitution group. Used as a
   kind of content unit within the information package map.
-->
<informationPackageMap xmlns="">
   <sip:sipContentUnit xmlns:sip="sip">
      <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>
      <!-- sipDescriptor and sipTemplate are part of specialized sipContentUnit -->
      <sip:sipDescriptor locType="URL" xlink:href="SPMS_00216.xml" />
      <sip:sipTemplate locType="URL" xlink:href="complexTemplateSIP.xml" />
   </sip:sipContentUnit>
</informationPackageMap>


<!-- Corresponding schema fragment-->
```

```
<xsd:element name="sipContentUnit" type="sipContentUnitType" substitutionGroup="xfdu:abstractContentUnit"/>
  <xsd:complexType name="sipContentUnitType">
    <xsd:complexContent>
      <xsd:extension base="xfdu:contentUnitType">
        <xsd:sequence>
          <xsd:element name="sipDescriptor" type="sipDescriptorLocationType"/>
          <xsd:element name="sipTemplate" type="sipTemplateLocationType"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="sipDescriptorLocationType">
    <xsd:complexContent>
      <xsd:extension base="xfdu:referenceType"/>
    </xsd:complexContent>
  </xsd:complexType>
  <xsd:complexType name="sipTemplateLocationType">
    <xsd:complexContent>
      <xsd:extension base="xfdu:referenceType"/>
    </xsd:complexContent>
  </xsd:complexType>
```

6.4 SEMANTICS

6.4.1 CONTENT UNIT TYPES (PROPOSED)

In this version of the recommendation, 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 listed below. It is anticipated that in future issues of this Recommendation this list may become an enumerated type and the semantics of each Content Unit type may be specified and enforced by Schematron-like mechanisms

6.4.1.1 Exchange Data Unit (EDU)

The Exchange Data Unit is a set of objects that has been packaged together for a reason. It is assumed that all the objects are related though a relationship that is not one of the predefined types defined by the units below.

6.4.1.2 Application Data Unit (ADU)

An Application Data Unit is a package type where all the objects are related to one or more primary content objects of interest

6.4.1.3 Description Data Unit (DDU)

A Description Data Unit is a package type where all the content objects are metadata objects intended to update the metadata repository of the consumer.

6.4.1.4 Process Description Unit (PDU)

A Process Description Unit can range from an automated scripting language to an English language description of the steps a person /intelligent agent would follow. The formal languages/schema used to define the PDU will be part of a future version.

7 INFORMATION PACKAGE MAP

7.1 OVERVIEW

The ***Information Package Map*** (informationPackageMap) 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

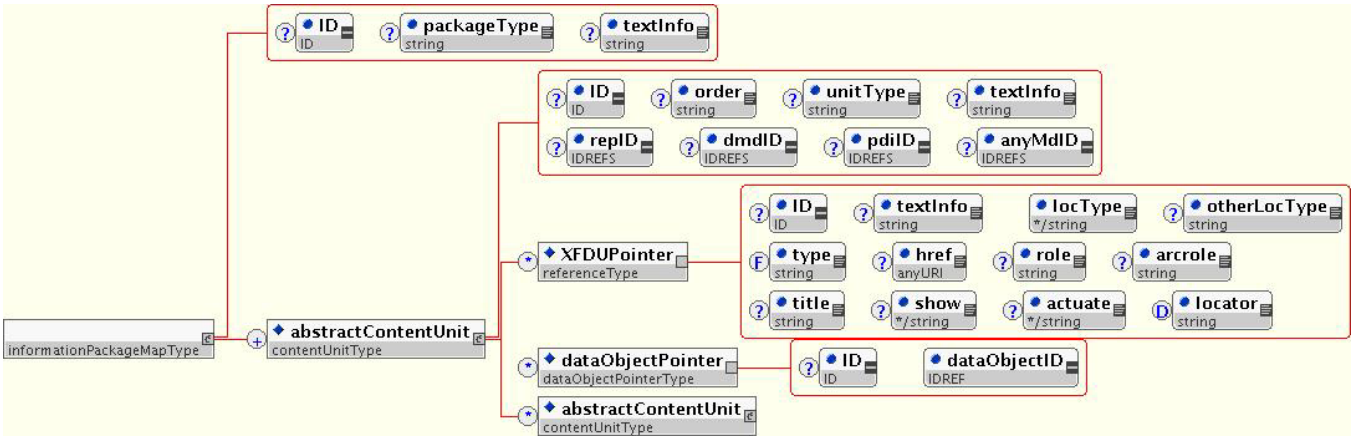


Figure 10: InformationPackageMapType

```
<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: ID: an XML ID for the element; TYPE: the type of
        Information Product provided. Typical values will be"AIP" for a
        map which describes a complete AIP obeying all constrainsts and
        cardinalities in the OAIS reference model "SIP" for a map which
        describes a Submission Information Package textInfo: a string to
        describe the informationPackageMap to users.
 packageType: a type for the object, e.g., book, journal, stereograph, etc.;
        Concrete implementation of contentUnit (defaultContentUnit, behavioralContentUnit,
etc) have to be used in the instance document.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element ref = "abstractContentUnit" maxOccurs = "unbounded"/>
    </xsd:sequence>
```

```
    <xsd:attribute name = "ID" type = "xsd:ID"/>
    <xsd:attribute name = "packageType" type = "xsd:string"/>
    <xsd:attribute name = "textInfo" type = "xsd:string"/>
  </xsd:complexType>
```

7.3 EXAMPLES

7.3.1 AN INFORMATION PACKAGE MAP

```
<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 OBJECT SECTION

8.1 OVERVIEW

A Data Object Section (dataObjectSection) contains one or more ***Data Object*** elements (dataObject). Each Data Object contains one ***Byte Stream***(byteStream) element that references or contains 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

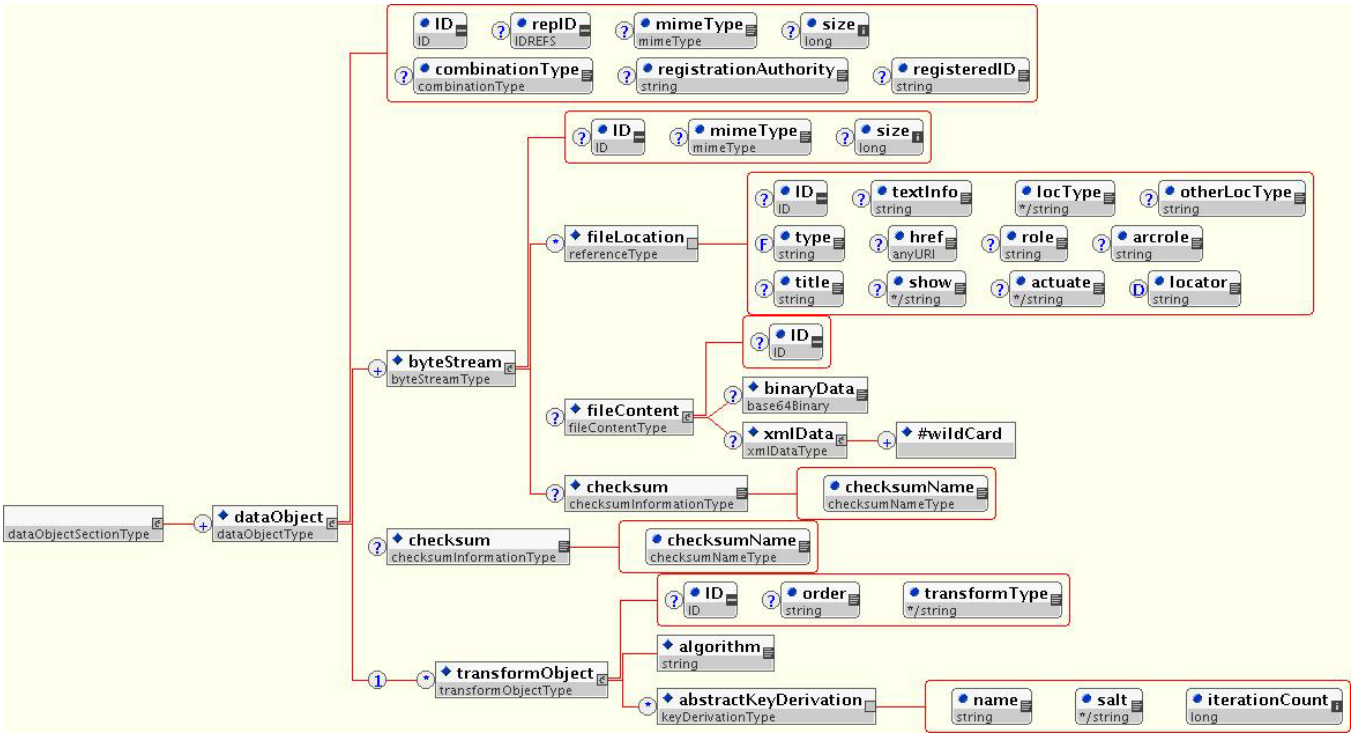


Figure 11: dataObjectType Schema Diagram

```
<xsd:complexType name = "keyderivationType">
<xsd:annotation>
  <xsd:documentation>key derivation type 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 default key derivation 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 key derivation in an instance of
      transformObject element would point not to instance of keyDerivation element, but rather 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="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 element
      that may contain an fileLocation element, which provides a pointer to
      a content byteStream, and/or an 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.
       repID list of representation metadata IDREFs. The size, checksum, and
      mime type are related to the original data before any transformations occured.
      5 combinationType - specifies if multiple byteStream objects are meant to be concatenated
      6 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:mimeType"/>
    <xsd:attribute name="size" type="xsd:long"/>
    <xsd:attribute name="combinationType" type="xfdu:combinationType" use="optional"/>
    <xsd:attributeGroup ref="xfdu:registrationGroup"/>
    </xsd:complexType>

  <xsd:simpleType name="combinationType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="concat"/>
    </xsd:restriction>
  </xsd:simpleType>

<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 element
      that may contain an fileLocation element, which provides a pointer to
      a content byteStream, and/or an 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.
       repID list of representation metadata IDREFs. The size, checksum, and
      mime type are related to the original data before any transformations occured.
      5 combinationType - specifies if multiple byteStream objects are meant to be concatenated
      6 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:mimeType"/>
    <xsd:attribute name="size" type="xsd:long"/>
    <xsd:attribute name="combinationType" type="xfdu:combinationType" use="optional"/>
    <xsd:attributeGroup ref="xfdu:registrationGroup"/>
    </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 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:mimeType"/>
  <xsd:attribute name="size" type="xsd:long"/>
  </xsd:complexType>
  <xsd:complexType name = "dataObjectSecType">
    <xsd:annotation>
      <xsd:documentation>dataObjectSecType : a container for one or more elements of dataObjectType
      </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name = "dataObject" type = "dataObjectType" maxOccurs = "unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
```

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 elementof dataObject element.

```
<!-- 13.3.1 Verify the checksum of the file - data object example-->
<dataObject repID = "mathMLAlgRepMD" ID = "mpeg21">
   <byteStream mimeType = "video/mpeg" ID = "mpeg21AnimData" size = "414131">
      <fileLocation locType = "URL" xmlns:ns1="http://www.w3.org/1999/xlink" ns1: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 mimeType attribute and checksum elementof the Data Object specify the mime type and checksum of the original data object, that is the bytestream before any transformations were applied. The values of the mimeType and checksum attributes of the byteString object are those of the current 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 locType = "URL" xmlns:ns2 = "http://www.w3.org/1999/xlink" ns2: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 an fileContent element and included physically inthe Manifest. The FileLocation element specifies anHTTP GET URL to request the latest version of data from an online registry/repository.

```
dataObject mimeType = "application/octetstream" ID = "orbitalData">
    <byteStream ID = "orbitData">
      <fileLocation locType = "URL" ns7:href = "http://coin.gsfc.nasa.gov:8080/ims-bin/3.0.1/nph-
  ims.cgi?msubmit=yes&amp;lastmode=SRCHFORM" xmlns:ns7 = "http://www.w3.org/1999/xlink"/>
      <fileContent>
        <binaryData>
          UEsDBBQACAAIAKqMBC8AAAAAAAAAAAAAAAAPAAAAeGZkdS8uY2xhc3NwYXRotZXfS8MwEMff/StK3
        5OugqCwH4hO0I...
        </binaryData>
      </fileContent>
      <checksum checksumName="CRC32">b3eb4b34</checksum>
    </byteStream>
  </dataObject>
```

8.4 SEMANTICS

These are the semantics for the existence of multiple FileLocation or FileComtent elements in a byteStream:

1. One FileComtent and one FileLocation means the FileComtent should serve as backup if the FileLocation is not accessable.

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 accessable.

3. Multiple FileLocations are undefined at this time the implementers are cautioned against defining semantics for this construct

4 One fileComtent and multiple FileLocation semantics are undefined at this timeare undefined at this time the implementers are cautioned against defining semantics for this construct. Note: Concatenating multiple physical files to form a single described object (eg, an image) is accomplished through the use of multiple bystestreams and the combinationtype attribute

9 METADATA SECTION TYPE AND METADATA OBJECTS

9.1 OVERVIEW

The ***Metadata Section*** (metadataSec) contains or references 0 or more ***Metadata Objects*** (metadataObjects) 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 aprovides 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', data entity dictionary ('DED'), 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 use 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 resouce, or 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 this description 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.

.

In the Metadata Object section, 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 2.

9.2 XML SCHEMA FOR METADATA OBJECTS

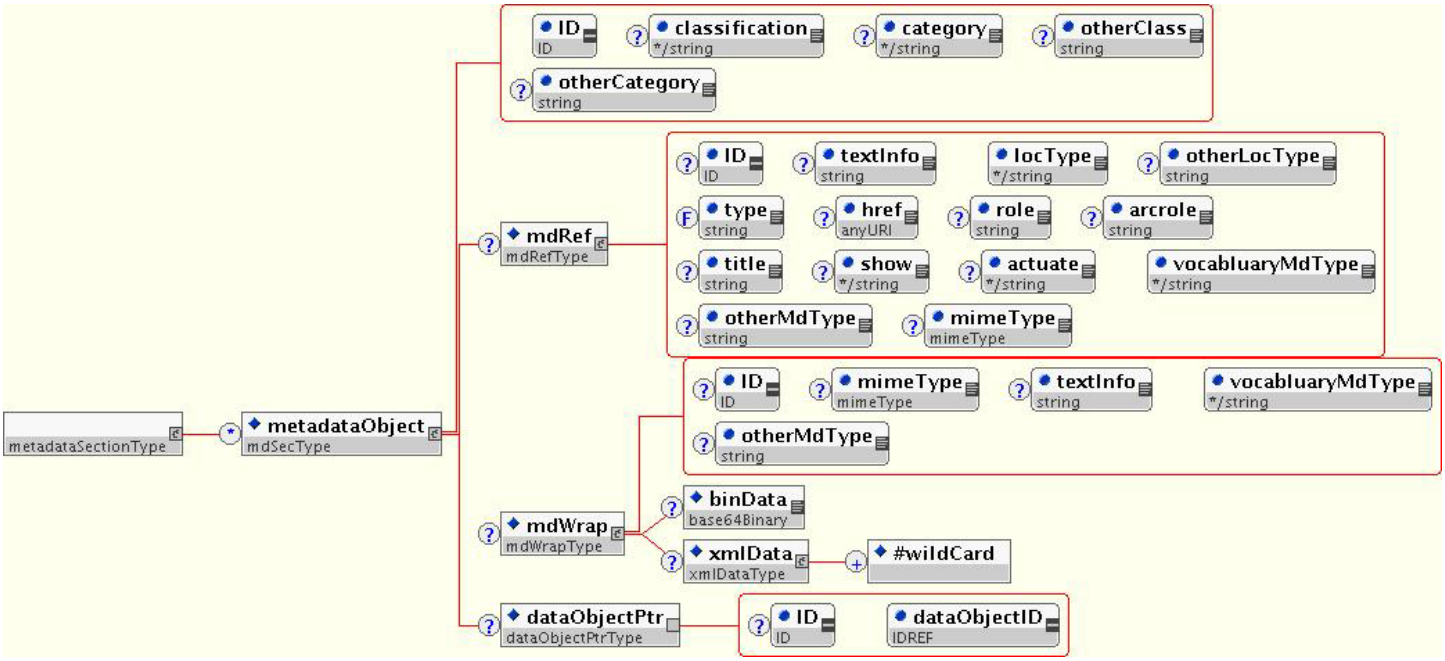


Figure 12: mdSecType and metadataSectionType Schema Diagram

```
<xsd:complexType " name = "mdSecType">
    <xsd:annotation>
      <xsd:documentation>mdSecType (metadata section) Complex Type A generic
      framework for pointing to/including metadata within a XFDU
      document, a la Warwick Framework. An mdSec 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 mdSecType
      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>
```

9.3 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 vocabularyMdType="OTHER" mimeType = "text/xml" textInfo = "spacecraft description" locType =
```

"URL" ns1:href = "file:packagesamples/scenario1/ecsdmd.xml" xmlns:ns1 = "http://www.w3.org/1999/xlink"/>
</metadataObject>
<metadataObject ID = "provenance" classification = "PROVENANCE" category = "PDI">
<metadataReference vocabularyMdType = "OTHER" mimeType = "text/xml" textInfo = "processing history XML file"
locType = "URL" ns1:href = "file:packagesamples/scenario1/pdi.xml" xmlns:ns1 = "http://www.w3.org/1999/xlink"/>
</metadataObject>
<metadataObject ID = "atdMD" classification = "OTHER" category = "REP">
<dataObjectPointer dataObjectID = "ATDMD"/>
</metadataObject>
</metadataSection>

10 BEHAVIOR SECTION AND BEHAVIOR OBJECTS

10.1 OVERVIEW

A ***Behavior Section*** (behaviorSection) contains one or more ***Behavior Objects*** (behaviorObject) that associate executable behaviors with content in the XFDU object. A Behavior Object contains an ***Interface Definition (interfaceDef)*** that represents an abstract definition of the set of behaviors represented by a particular Behavior Object. A Behavior Object also may contain a ***Mechanism*** that is a module of executable code that implements and runs the behaviors defined abstractly by the interface definition. In the 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.

The ability to specify Interfaces Definitions and implementation dependant mechanisms is a powerful tool. However, the full potential of these feature will be realized when the value of the content unit can be the result of the application of Behaviors to data in the content unit. The ability to monitor and control the application of behaviors to data objects is a major goal of the XFDU Version 2 Recommendations.

10.2 XML SCHEMA FOR BEHAVIOR OBJECTS

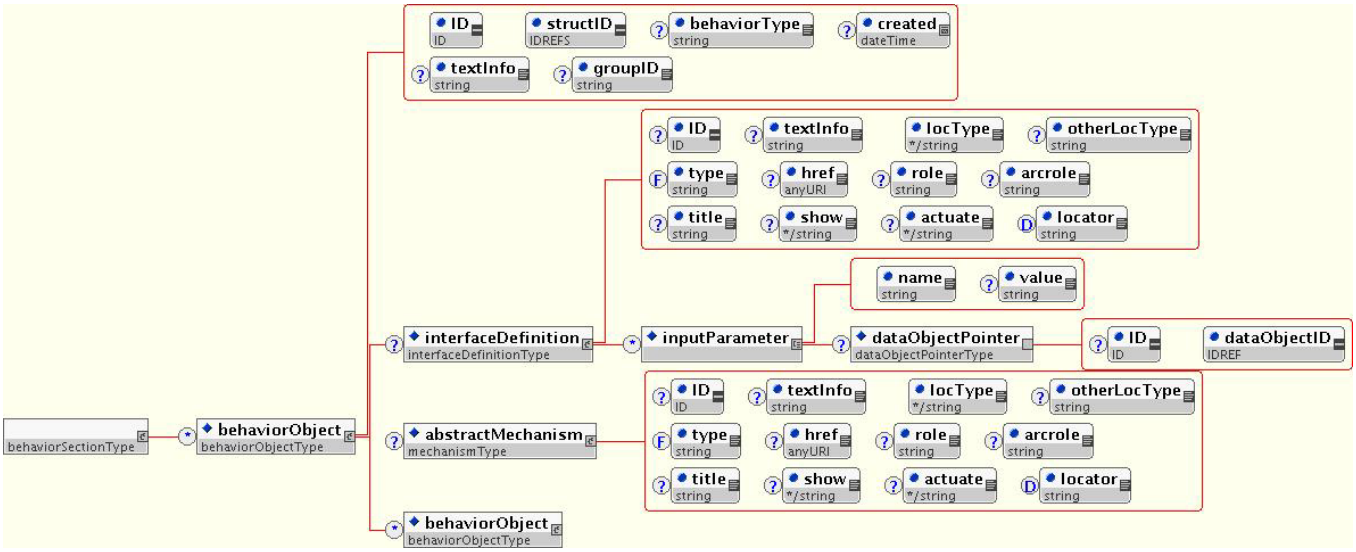


Figure 13: behaviorObjectType Schema Diagram

```
<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 section 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 section may have the following
        attributes: 1. ID: an XML ID for the element 2. structID: IDREFS
        to information package map sections or divs within a information package map in the XFDU
        document.
        3. behaviorType: a behavior type
        identifier for a given set of related behaviors. 4. created:
        date this behavior section of the XFDU object was created. 5.
        textInfo: a description of the type of behaviors this section
        represents. 6. groupID: an identifier that establishes a
        correspondence between this behavior section and behavior
        sections. Typically, this will be used to facilitate versioning
        of behavior sections.  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" minOccurs="0"/>
      <xsd:element ref="xfdu:abstractMechanism" minOccurs="0"/>
      <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="structID" 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: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="behaviorObjectType">
    <xsd:annotation>
      <xsd:documentation>behaviorObjectType: Complex Type for Behaviors. A
        behavior section 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 section may have the following
        attributes: 1. ID: an XML ID for the element 2. structID: IDREFS
        to information package map sections or divs within a information package map in the XFDU
```

```
        document.
        3. behaviorType: a behavior type
        identifier for a given set of related behaviors. 4. created:
        date this behavior section of the XFDU object was created. 5.
        textInfo: a description of the type of behaviors this section
        represents. 6. groupID: an identifier that establishes a
        correspondence between this behavior section and behavior
        sections. Typically, this will be used to facilitate versioning
        of behavior sections.  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" minOccurs="0"/>
      <xsd:element ref="xfdu:abstractMechanism" minOccurs="0"/>
      <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="structID" 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 = "behaviorSectionType">
      <xsd:sequence>
        <xsd:element name = "behaviorObject" type = "behaviorObjectType" minOccurs = "0" maxOccurs = "unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
    </xsd:complexType>
```

10.3 EXAMPLES

10.3.1 EXAMPLE OF DEFINING AN INTERFACE AND PARAMETER

```
<behaviorSection>
   <behaviorObject ID="behaviorObject" structID="cu1">
      <interfaceDefinition locType="URL" xmlns:ns3="http://www.w3.org/1999/xlink"
ns3:href="http://sindbad.gsfc.nasa.gov/processmpg21.wsdl">
         <inputParameter name="mpeg21Input">
            <dataObjectPointer dataObjectID="mpeg21"/>
         </inputParameter>
      </interfaceDefinition>
   </behaviorObject>

</behaviorSection>
```

10.4 SEMATICS

These semantics will be defined during the implementation activities and included prior to the final version of this Recommendation

11 FULL XML SCHEMA –NORMATIVE/RULING

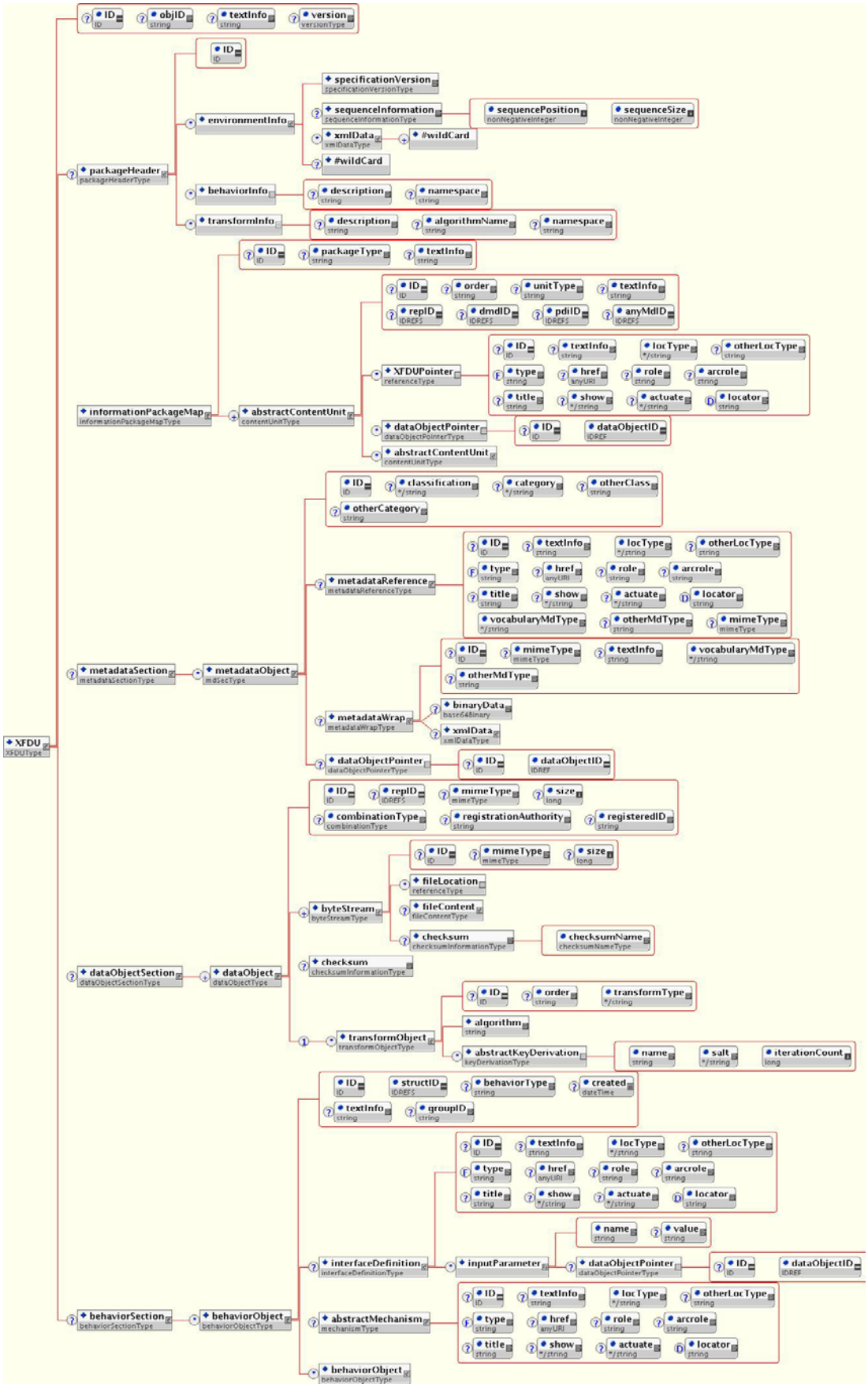

XFDU
XFDUType
ID
objID
textInfo
version
packageHeader
packageHeaderType
environmentInfo
specificationVersion
sequenceInformation
sequencePosition
sequenceSize
xmlData
#wildCard
behaviorInfo
description
namespace
transformInfo
algorithmName
informationPackageMap
informationPackageMapType
abstractContentUnit
contentUnitType
packageType
order
unitType
replID
dmdID
pdiID
anyMdID
XFDUPointer
referenceType
dataObjectPointer
dataObjectPointerType
dataObjectID
metadataSection
metadataSectionType
metadataObject
mdSecType
classification
otherClass
otherCategory
metadataReference
metadataReferenceType
locType
otherLocType
type
href
role
arcrole
title
show
actuate
locator
vocabularyMdType
otherMdType
mimeType
metadataWrap
binaryData
xmlData
dataObjectSection
dataObjectSectionType
dataObject
dataObjectType
combinationType
registrationAuthority
registeredID
size
byteStream
fileLocation
fileContent
checksum
checksumNameType
transformObject
transformObjectType
transformType
algorithm
abstractKeyDerivation
name
salt
iterationCount
behaviorSection
behaviorSectionType
behaviorObject
behaviorObjectType
structID
behaviorType
created
groupID
interfaceDefinition
interfaceDefinitionType
inputParameter
value
abstractMechanism
mechanismType

Figure 14 Full XFDU Schema Diagram

11.1 FULL XML SCHEMA

```
<?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="http://www.ccsds.org/xfdu/2004" xmlns:xlink="http://www.w3.org/1999/xlink"
targetNamespace="http://www.ccsds.org/xfdu/2004" elementFormDefault="unqualified" attributeFormDefault="unqualified">
  <xsd:import namespace="http://www.w3.org/1999/xlink" schemaLocation="xlink.xsd"/>
  <xsd:attributeGroup name="METADATA">
    <xsd:annotation>
      <xsd:documentation>
        This attribute group aggregates attributes that can be used for specifying metadata type
        This group includes following attributes:
        vocabularyMdType specifies metadata type (e.g. MARC.DDI)
        otherMdType specifies metadata type in case vocabularyMdType has value of OTHER
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="vocabularyMdType" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="MARC"/>
          <xsd:enumeration value="EAD"/>
          <xsd:enumeration value="DC"/>
          <xsd:enumeration value="FGDC"/>
          <xsd:enumeration value="OTHER"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="otherMdType" type="xsd:string"/>
  </xsd:attributeGroup>
  <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:
        locType specifies location type (e.g. URN,URL)
        otherLocType specifies location type in case locType has value of OTHER
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="locType" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="URL"/>
          <xsd:enumeration value="OTHER"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="otherLocType" type="xsd:string"/>
  </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
        registrationId - the id for the registration
      </xsd:documentation>
    </xsd:annotation>
    <xsd:attribute name="registrationAuthority" use="optional"/>
    <xsd:attribute name="registeredID" type="xsd:string" use="optional"/>
```

```
</xsd:attributeGroup>
<xsd:simpleType name="versionType">
  <xsd:annotation>
    <xsd:documentation>
      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:restriction base="xsd:string"/>
</xsd:simpleType>
<xsd:simpleType name="mimeType">
  <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="combinationType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="concat"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:attribute name="namespace" type="xsd:string"/>
<xsd:simpleType name="sequenceIdentifierType">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>
<xsd:complexType name="sequenceInformationType">
  <xsd:annotation>
    <xsd:documentation>
      An element of this type encapsulates information about possible sequence of XFDU package
      which form some logical XFDU unit. sequenceInformationType has two mandatory attributes:
      1. sequencePosition - the position of this XFDU package in the sequence; if 0 is specified
       and sequenceSize is unknow 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:attribute name="ID" type="xsd:ID"/>
  <xsd:attribute name="textInfo" type="xsd:string"/>
  <xsd:attributeGroup ref="xfdu:LOCATION"/>
  <xsd:attributeGroup ref="xlink:simpleLink"/>
  <xsd:attribute name="locator" type="xsd:string" use="optional" default="/"/>
</xsd:complexTyp
<xsd:complexType name="checksumInformationType">
  <xsd:annotation>
    <xsd:documentation>
      An element of this type would convey checksum information: checksum value and type 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="mdSecType">
    <xsd:annotation>
      <xsd:documentation>mdSecType (metadata section) Complex Type A generic
      framework for pointing to/including metadata within a XFDU
      document, a la Warwick Framework. An mdSec 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 mdSecType
      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>
        Entity of this type is used to indicate CCSDS document identifier of the issue of XFDU specification on which this
        XFDU is based
      </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 section consists of three possible subsidiary
    sections: environmentInfo (specification of the hardware and software
    platform which created this package), behaviorInfo and transformInfo (the names, classifications, parameter
    names/types and any other information needed to reverse
    trans formations used in the XFDU). packageHeaderType has a single
    attribute, ID: an XML ID.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <xsd:element name="environmentInfo" minOccurs="0" maxOccurs="unbounded">
      <xsd:annotation>
        <xsd:documentation>environmentInfo: technical metadata. The environmentInfo element
        provides a wrapper around a generic metadata section that
        should contain technical metadata regarding a dataObject or dataObjects. It
        has an attribute, specVersion, which specifies XFDU version for which this XFDU package was created.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
        <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:element name="xmlData" type="xfdu:xmlDataType" minOccurs="0" maxOccurs="unbounded"/>
          <xsd:any namespace="##other" minOccurs="0" processContents="strict"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="behaviorInfo" minOccurs="0" maxOccurs="unbounded">
      <xsd:annotation>
        <xsd:documentation>
          behaviorInfo contains:
          description - general description
          namespace - namespace of the specified technology
          behaviorInfo has an optional xmlData element to include any additional metadata if needed
        </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
        <xsd:attribute name="description" type="xsd:string"/>
        <xsd:attribute ref="xfdu:namespace"/>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="transformInfo" minOccurs="0" maxOccurs="unbounded">
      <xsd:annotation>
        <xsd:documentation>transformInfo (the names, classifications, parameter
        names/types and any other information needed to reverse
        transformations used in the XFDU)
        transformInfo contains:
        description - general description
        algorithmName -name of transformation algorithm
        namespace - namespace of the specified technology
        transformInfo has optional xmlData element to include any additional metadata if needed
        </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
        <xsd:attribute name="description" type="xsd:string"/>
        <xsd:attribute name="algorithmName" type="xsd:string"/>
        <xsd:attribute ref="xfdu:namespace"/>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="ID" type="xsd:ID" use="required"/>
</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. locType: the
    type of locater contained in the body of the element; 3.
    otherLocType: a string indicating an alternative locType when
    the locType attribute value is set to "OTHER."; 4. xlink:href:
    see XLink standard (http://www.w3.org/TR/xlink) 5. xlink:role:
    "" 6. xlink:arcrole: "" 7. xlink:title: "" 8. xlink:show: "" 9.
    xlink:actuate: "" 10. mimeType: the MIME type for the metadata
    being pointed at; 11. vocabularyMdType: the type of metadata being pointed
    at (e.g., MARC, EAD, etc.); 12. 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 xlink:href attribute,
    supplemented by the XPTR attribute as needed.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
    <xsd:extension base="xfdu:referenceType">
      <xsd:attributeGroup ref="xfdu:METADATA"/>
      <xsd:attribute name="mimeType" type="xfdu:mimeType"/>
    </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. vocabularyMdType: 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:mimeType"/>
      <xsd:attribute name="textInfo" type="xsd:string"/>
      <xsd:attributeGroup ref="xfdu:METADATA"/>
    </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>key derivation type 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
    default key derivation 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 key derivation in an instance of
    transformObject element would point not to instance of keyDerivation element, but rather 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 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:mimeType"/>
    <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 element
      that may contain an fileLocation element, which provides a pointer to
      a content byteStream, and/or an 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 dataObjectin bytes
        4. checksum: a checksum for dataObject. Checksum information provided via option checksum element.
        5. repID list of representation metadata IDREFs. NOTE: The size, checksum, and mime type are related to the
           original data before any transformations occured.
        6.  combinationType - specifies if multiple byteStream objects are meant to 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:mimeType"/>
    <xsd:attribute name="size" type="xsd:long"/>
    <xsd:attribute name="combinationType" type="xfdu:combinationType" 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 content
      units element has the following attributes:
      1.ID (an XML ID);
      2.order: an 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 metdata related to this  contentUnit
    8.unitType: a type of content unit (e.g., Application
    Data Unit, Data Description Unit, Software Installation Unit, etc.).
    contentUnitType is declared as a base type for concrete implementations of contentUnit.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:sequence>
    <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: ID: an XML ID for
        this element locType: the type of locater contained in the
        xlink:href attribute; otherLocType: a string to indicate an
        alternative locType if the locType attribute itself has a value
        of "OTHER." xlink:href: see XLink standard
        (http://www.w3.org/TR/xlink) xlink:role: "" xlink:arcrole: ""
        xlink:title: "" xlink:show: "" xlink:actuate: "" NOTE: XFDUPointer
        is an empty element. The location of the resource pointed to
        MUST be stored in the xlink:href element.
        </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: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: ID: an XML ID for the element; TYPE: the type of
        Information Product provided. 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 textInfo: a string to
        describe the informationPackageMap to users. packageType: a type for the object, e.g., book, journal, stereograph, etc.;

        Concrete implementation of abstractContentUnit (contentUnit etc) have to 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: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 section 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 section may have the following
        attributes: 1. ID: an XML ID for the element 2. structID: IDREFS
        to information package map sections or divs within a information package map in the XFDU
        document.
        3. behaviorType: a behavior type
        identifier for a given set of related behaviors. 4. created:
        date this behavior section of the XFDU object was created. 5.
        textInfo: a description of the type of behaviors this section
        represents. 6. groupID: an identifier that establishes a
        correspondence between this behavior section and behavior
        sections. Typically, this will be used to facilitate versioning
        of behavior sections.  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" minOccurs="0"/>
      <xsd:element ref="xfdu:abstractMechanism" minOccurs="0"/>
      <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="structID" 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:mdSecType" 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: an 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"/>
```

Annex A Complete Example XFDU

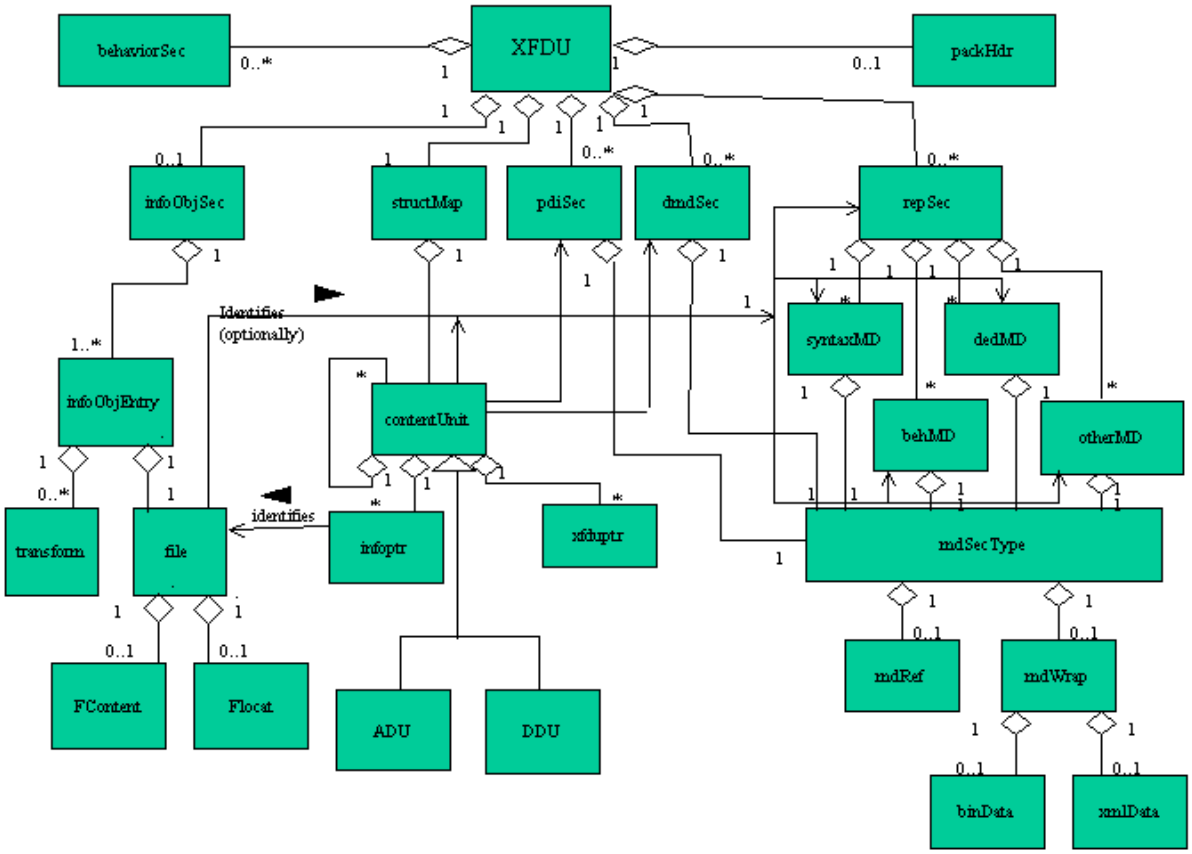
```
<?xml version="1.0" encoding="UTF-8"?>
<xfdu:XFDU xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ccsds.org/xfdu/2004"
 xmlns:ns1="xfdu:http://www.ccsds.org/xfdu/2004"
 xmlns:xfdu="http://www.ccsds.org/xfdu/2004">
    <packageHeader ID="packageHeader">
      <environmentInfo>
         <specificationVersion>1.0</specificationVersion>
         <!-- sequence information attribute is specified by producer-->
         <sequenceInformation sequenceSize="10" sequencePosition="1">producer1.seq10</sequenceInformation>
         <xmlData>
            <platform>Linux2.4.22-1.2129.nptl</platform>
         </xmlData>
      </environmentInfo>
      <transformInfo algorithmName = "blowfish" description = "encryption"/>
   </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 vocabularyMdType="OTHER" mimeType = "text/xml" textInfo = "spacecraft description" locType =
"URL" ns1:href = "file:packagesamples/scenario1/ecsdmd.xml" xmlns:ns1 = "http://www.w3.org/1999/xlink"/>
      </metadataObject>
      <metadataObject ID = "provenance" classification = "PROVENANCE" category = "PDI">
         <metadataReference vocabularyMdType = "OTHER" mimeType = "text/xml" textInfo = "processing  history  XML file"
locType = "URL" ns1:href = "file:packagesamples/scenario1/pdi.xml" xmlns:ns1 = "http://www.w3.org/1999/xlink"/>
      </metadataObject>
      <metadataObject ID = "atdMD" classification = "OTHER" category = "REP">
         <dataObjectPointer dataObjectID = "ATDMD"/>
      </metadataObject>
      <!--This metadata is categorized as REP (representation) and classified as OTHER-->
      <metadataObject ID = "mathMLAlgRepMD" classification = "OTHER" category = "REP">
         <metadataWrap vocabularyMdType = "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 locType = "URL" xmlns:ns1="http://www.w3.org/1999/xlink" ns1: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 locType = "URL" xmlns:ns2 = "http://www.w3.org/1999/xlink" ns2: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 locType = "URL" ns7:href = "http://coin.gsfc.nasa.gov:8080/ims-bin/3.0.1/nph-
ims.cgi?msubmit=yes&amp;lastmode=SRCHFORM" xmlns:ns7 = "http://www.w3.org/1999/xlink"/>
            <fileContent>

<binaryData>UEsDBBQACAAIAKqMBC8AAAAAAAAAAAAAAAAPAAAAeGZkdS8uY2xhc3NwYXRotZXfS8MwEMff/StK35Oug
qCwH4hO0I...</binaryData>
            </fileContent>
            <checksum checksumName="CRC32">b3eb4b34</checksum>
         </byteStream>
      </dataObject>
      <dataObject repID = "atdMD" ID = "hdfFile0">
         <byteStream mimeType = "application/x-hdf" ID = "hdfFile0bs" size = "10455471">
            <fileLocation locType = "URL" ns4:href = "file:packagesamples/scenario1/mod1.hdf" xmlns:ns4 =
"http://www.w3.org/1999/xlink"/>
            <checksum checksumName="CRC32">acab6535</checksum>
```

```
         </byteStream>
      </dataObject>
      <dataObject repID = "atdMD" ID = "hdfFile1">
         <byteStream mimeType = "application/x-hdf" ID = "hdfFile1bs" size = "10455471">
            <fileLocation locType = "URL" ns5:href = "file:packagesamples/scenario1/mod2.hdf" xmlns:ns5 =
"http://www.w3.org/1999/xlink"/>
            <checksum checksumName="CRC32">acab6535</checksum>
         </byteStream>
      </dataObject>
      <dataObject repID = "atdMD" ID = "hdfFile2">
         <byteStream mimeType = "application/x-hdf" ID = "hdfFile2bs" size = "10455471">
            <fileLocation locType = "URL" ns6:href = "file:packagesamples/scenario1/mod3.hdf" xmlns:ns6 =
"http://www.w3.org/1999/xlink"/>
            <checksum checksumName="CRC32">acab6535</checksum>
         </byteStream>
      </dataObject>
   </dataObjectSection>
   <behaviorSection>
      <behaviorObject ID="behaviorObject" structID="cu1">
         <interfaceDefinition locType="URL" xmlns:ns3="http://www.w3.org/1999/xlink"
ns3:href="http://sindbad.gsfc.nasa.gov/processmpg21.wsdl">
            <inputParameter name="mpeg21Input">
               <dataObjectPointer dataObjectID="mpeg21"/>
            </inputParameter>
         </interfaceDefinition>
      </behaviorObject>
   </behaviorSection>
```

Annex B UML for XFDU – *****DON TO DO



Annex C Legend for XML Authority Figures

 - indicates that attribute or element is optional

 - indicates 0 to many occurrences

 - indicates 1 to many occurrences

 - indicates fixed attribute

 - indicates choice

 - indicates that item has children

 - indicates 1 to many relationship between elements

 - indicates that an attribute has default value