Consultative
Committee for
Space Data Systems

DRAFT RECOMMENDATION FOR SPACE
DATA SYSTEM STANDARDS

Producer-Archive Interface Specification

CCSDS 651.1-W-05

DRAFT White BOOK

September 2006

AUTHORITY

Issue:	White book, Issue 5
Date:	September 2006
Location:	

(WHEN APPROVED FOR PUBLICATION AS A RECOMMENDATION, THE FOLLOWING TEXT WILL APPEAR)

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 authorisation of CCSDS Recommendations is detailed in Reference [B1], and the record of Agency participation in the authorisation of this document can be obtained from the CCSDS Secretariat at the address below.

This Recommendation is published and maintained by:

CCSDS Secretariat
Program Integration Division (Code-M3)
National Aeronautics and Space Administration
Washington, DC 20546, USA

STATEMENT OF INTENT

(WHEN APPROVED FOR PUBLICATION AS A RECOMMENDATION, THE FOLLOWING TEXT WILL APPEAR)

The Consultative Committee for Space Data Systems (CCSDS) is an organization officially established by the management of the member space Agencies. The committee meets periodically to address data system 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 the committee 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 **Recommendations** is entirely voluntary. Endorsement, however, indicates the following understandings:

- Whenever an Agency establishes a CCSDS-related **standard**, this **standard** will be in accordance 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 shall be made via memorandum 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 cancelled.

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

FOREWORD

(WHEN THIS RECOMMENDATION IS FINALIZED, IT WILL CONTAIN THE FOLLOWING FOREWORD:)

This Recommendation is a technical Recommendation providing the abstract syntax and an XML implementation for the Plan of Transfer, the Submission Information Package (SIP) Model to fulfil parts of the process defined in the *Producer Archive Ingest Methodology Abstract Standard (PAIMAS)* [Ref 1].

Through the process of normal evolution, it is expected that expansion, deletion or modification to this document may occur. This Recommendation is therefore subject to CCSDS document management and change control procedures. Current versions of CCSDS documents are maintained at the CCSDS Web site:

http://www.ccsds.org/

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

DOCUMENT CONTROL

Document	Title	Date	Status/Remarks
CCSDS 651.1-W-0.1	Recommendation for Space Data System Standards: Specification for the Formal Definition and Transfer Phase of a Producer-Archive Interface	September 2004	Original Draft
CCSDS 651.1-W-0.2	Recommendation for Space Data System Standards: Producer-Archive Interface Specification	December 2005	Original Draft
CCSDS 651.1-W-0.3	Recommendation for Space Data System Standards: Producer-Archive Interface Specification	February 2006	
CCSDS 651.1-W-0.4	Recommendation for Space Data System Standards: Producer-Archive Interface Specification	April 2006	Updates to Descriptor content

CONTENTS

Section

1 INTRODUCTION

1.1 PURPOSE AND SCOPE

The purpose of this recommendation is to provide a standard method to formally define the digital information objects to be transferred by an information Producer to an Archive and for effectively transferring these objects in the form of **Submission Information Packages (SIPs).**

This recommendation fits into the context defined by:

- The 'Producer Archive Interface Methodology Abstract Standard' recommendation (PAIMAS) (See reference [1]).
- The 'Reference Model for an Open Archival Information System' recommendation (OAIS) (See reference [2]).
- The 'XML Formatted Data Unit (XFDU), Structure and Construction Rules' draft White Book (see reference [5]).

The PAIMAS Recommendation (See Reference [1]) defines a methodology based on the four following phases: Preliminary, Formal Definition, Transfer, Validation.

This Recommendation applies specifically to the implementation of the main part of the Formal Definition Phase and the Transfer Phase, taking into account part of the Validation Phase.
The proposed implementation should help in the automation and the management of the Transfer and Validation Phases.

The proposed method may however be used, to some extent, for the Preliminary Phase.

This Recommendation does not exclude other PAIMAS implementation Recommendations.

1.2 APPLICABILITY

The implementation defined in this document applies to any Producer-Archive Project. It is specifically applicable to those organizations and individuals who create information that may need Long-Term Preservation and with organizations making information available for the Long Term.

This application is relevant only if both partners in the Producer-Archive project agree with **a** shared implementation as defined in this document.

1.3 RATIONALE

This recommendation aims at overcoming the many difficulties encountered during transactions between information Producers and the Archives.

Regarding the Formal Definition Phase, this Recommendation should enable:

- the Producer to have a very precise, unambiguous definition of the different Digital Objects to be produced, of the form and possibly the order in which they should be delivered,
- the Archive to be sure that the Digital Objects which are to be transferred to it will enable it to build **Archival Information Packages** which have all of the characteristics defined in the OAIS Reference Model,
- the respective Managers of the Producer and the Archive to be fully aware of all details of their commitments in terms of human and financial resources.

Regarding the Transfer Phase, this Recommendation should enable a high degree of automation and verification of the transfer process (recognize the schedule for the Data Submission Sessions, guarantee that the operation runs well technically, etc.).

Regarding the Validation Phase, this Recommendation should enable the use of tools for systematically validating that the objects received are those expected, and that they conform to the level of detail previously agreed.

1.4 CONFORMANCE

To be completed.

1.5 DOCUMENT STRUCTURE

To be completed later.

1.6 DEFINITIONS

1.6.1 ACRONYMS AND ABBREVIATIONS

This sub-section defines the acronyms and abbreviations which are used throughout this Recommendation:

AIP	Archival Information Package
ASCII	American Standard Code for Information Interchange
CCSDS	Consultative Committee for Space Data Systems
DO	Data Object
DTD	Document Type Definition
EAD	Encoded Archival Description
EAST	Enhanced Ada Subset
ID	Identifier
ISO	International Organization for Standardization
OAIS	Open Archival Information System
PAIMAS	Producer Archive Interface Methodology Abstract Standard

PDI	Preservation Description Information
PDF	Portable Document Format
POT	Plan of Objects to be Transferred
RM	Reference Model
SIP	Submission Information Package
UML	Unified Modelling Language
XFDU	XML Formatted Data Units
XML	eXtensible Markup Language

1.6.2 GLOSSARY OF TERMS

Following is a short glossary of the OAIS terminology indispensable for this document. The terminology used is fully defined in references [2] and [1], except the definitions printed in italics. Only brief definitions are provided here. This terminology does not seek to replace existing terminology in the various domains related to archiving. Each domain should be able to apply this methodology while retaining their specific terminology.
When first used in the text, the terms defined in the terminology are shown in bold.

Moreover, we assume that it is not necessary for the Producer to know and understand the information model and the typology of the OAIS information categories in detail, for instance Content Information, Representation Information, Preservation Description Information, etc. Indeed, it is the Archive's task to create AIPs from the SIPs transferred and thus to establish the suitable link between a given object coming from the Producer and any particular information category in the AIP within which this object will be inserted. To establish a dialog, the Producer and the Archive must agree on a common terminology and a common understanding of the associated concepts.

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

Archive: An organization that intends to preserve information for access and use by a Designated Community.

***Collection Descriptor**: describes a collection of one or more Transfer Object Types and its data objects relationship to other data objects whether represented by other Transfer Object Descriptors or Collection Descriptors.*

Consumer: The role played by those persons, or client systems, who interact with OAIS services to find preserved information of interest and to access that information in detail. This can include other OAISs, as well as internal OAIS persons or systems.

Data Object: Either a Physical Object or a Digital Object.

Data Submission Session: A delivered set of media or a single telecommunications session that provides data to an OAIS. The Data Submission Session format/contents is based on a data model negotiated between the OAIS and the Producer in the Submission Agreement. This data model identifies the logical constructs used by the Producer and how these are represented on each media delivery or in the telecommunication session.

***Descriptor**: An information unit used to describe a Transfer Object Type or one or more collections of Transfer Object Types, and to specify relationships among the data objects corresponding to those types.*

***Descriptor Model:** a model that defines the mandatory and optional attributes needed for a Descriptor.*

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.

EAST: The EAST language is a CCSDS and ISO norm. EAST offers a means to describe the syntax of a data file, including:

– the fields in which it can be decomposed;

– structure (simple or composite);

– type (integer, real, enumerated, array, record, list);

– range (min value, max value);

– coding (ASCII, binary);

– location (rank, length);

– optionality (mandatory or not and, if not, presence condition);

– eventually, variable dimension (for arrays).

Fixity Information: The information which documents the authentication mechanisms and provides authentication keys to ensure that the Content Information Object has not been altered in an undocumented manner.

Identifier : An XML CDATA (character data), that designates something (from DEDSL).

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 Package: The Content Information and associated Preservation Description Information which is needed to aid in the preservation of the Content Information.

Ingest: The OAIS entity that contains the services and functions that accept Submission Information Packages from Producers, prepares Archival Information Packages for storage, and ensures that Archival Information Packages and their supporting Descriptive Information become established within the OAIS.

Meta-data: Data about the content, the quality, condition and other characteristics of the data (from the FGDC Standards Reference Model (see [B5])).

Model : A data entity described independently from any instance in a data product and corresponding to a re-usable data entity definition from which other data entities may inherit the attributes and apply some specialization rule. (from DEDSL).

Packaging Information: The information that is used to bind and identify the components of an Information Package. For example, it may be the ISO 9660 volume and directory information used on a CD-ROM to provide the content of several files containing Content Information and Preservation Description Information.

Plan of Objects to be Transferred (POT): *Composed of the associated set of Descriptors, it gives a logical and hierarchical representation of the objects to be transferred for a given Producer-Archive Project. It must provide a global and understandable view of the objects of the project, and the relationships among them. It has the form of a tree. The objects to be transferred, called 'Transfer Objects' are represented by the leaves of the POT. Thus, the nodes of the POT have a different meaning depending on whether they are leaves or not:*

- *a leaf node corresponds to a single Transfer Object Type and therefore one exists for each Transfer Object Descriptor.*
- *A non-leaf node corresponds to a collection view of Transfer Object Types, including collections of collections. A non-leaf node exists for each Collection Descriptor.*

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.

Producer: The role played by those persons or client systems who provide the information to be preserved. This can include other OAISs or internal OAIS persons or systems.

Producer-Archive Project: A Producer-Archive Project is a set of activities and the means used by the information Producer as well as the Archive to ingest a given set of information into the Archive.

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.

Submission Agreement: The agreement reached between an OAIS and the Producer that specifies a data model for the Data Submission Session. This data model identifies format/contents and the logical constructs used by the Producer and how they are represented on each media delivery or in a telecommunication session. In the framework of this abstract methodology, the Submission Agreement will also deal with other aspects such as validation, change management and schedule.

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.

Transfer: The act involved in a change of physical custody of SIPs. This definition is derived from the International Council on Archives [ICA] Dictionary on Archival Terminology (see [B6]).

Transfer Object *: A structured and organized set of one or more Data Objects that are to be transferred to the*

Archive. There may be multiple Transfer Objects of the same Transfer Object Type.

***Transfer Object Type** : A set of characteristics describing a Transfer Object (such as the size of this object, the description of its content, and its makeup in terms of one or more data object types) .*

***Transfer Object Descriptor**: An information unit used to describe the Transfer Object Type and its data objects relationship to other data objects.*

1.7 REFERENCES

The following documents contain provisions (through references within this text) which constitute provisions of this Recommendation. At the time of the publication the indicated editions were valid. All documents are subject to revision, and users of this Recommendation are encouraged to investigate the possibility of applying the most recent editions of the documents indicated below. The CCSDS Secretariat maintains a register of currently available CCSDS Recommendations.

[1] *Producer Archive Interface methodology Abstract Standard* Recommendation for Space Data System Standards, CCSDS 651.0-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, May 2004. [Equivalent to ISO 20652:2006.]

[2] *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. [Equivalent to ISO 14721:2003.]

[3] Extensible Markup Language (XML) *1.0 (Third Edition) - W3C Recommendation 6 February 2004. http://www.w3.org/TR/2004/REC-xml-20040204/*

[4] XML Schema specification, part 1 (structure) and part 2 (data types) - W3C Recommendation 2 May 2001 http://www.w3.org/XML/Schema#dev

[5] *XML Formatted Data Unit (XFDU), Structure and Construction Rules*, CCSDS White Book, Washington, D.C.: CCSDS, 23 August 2006.

2 GENERAL FRAMEWORK

2.1 THE PRODUCER-ARCHIVE INTERFACE METHODOLOGY

The general context for this recommendation is that of the transfer of Digital Objects from a data Producer to an Archive. The methodology for specifying, performing and validating this transfer is defined in the PAIMAS Recommendation (see Reference [1]).

The term 'Producer' designates the persons who, and systems which, supply the Archive with information to be preserved. The Archive is an OAIS Archive. The main responsibility of an Archive is to preserve information and to make it available, in an intelligible and usable form, to a defined Designated Community.

Both Producer and Archive are assumed to be involved in a **Producer-Archive Project**: A Producer-Archive Project refers to both a set of activities and the means used by the information Producer as well as the Archive to ingest a given set of information into the Archive.

The Producer-Archive interactions, in a given Producer-Archive Project, consist of four different phases:

– The Preliminary Phase, also known as a pre-ingest or pre-accessioning phase, includes the initial contacts between the Producer and the Archive and any resulting feasibility studies, preliminary definition of the scope of the project, a draft of the **Submission Information Package** (SIP) definition and finally a draft **Submission Agreement**.

– The Formal Definition Phase includes completing the SIP design with precise definitions of the Digital Objects to be delivered, completing the Submission Agreement with precise, contractual **transfer** conditions such as restrictions on access and establishing the delivery schedule.

– The Transfer Phase performs the actual transfer of the SIP from the Producer to the Archive and the preliminary processing of the SIP by the Archive, as defined in the agreement.

– The Validation Phase includes the actual validation processing of the SIP by the Archive and any required follow-up action with the Producer. Different systematic or in-depth levels of validation may be defined. Validations may be performed after each delivery, or later, depending on the validation constraints.

Phases are carried out in chronological order. However, the Transfer Phase may overlap the Validation Phase. Each phase is divided into a number of sub-phases that must also be carried out in chronological order. Each of these sub-phases is made up of one or more action tables. The action tables and the actions can be carried out in any order.

2.2 THE FORMAL DEFINITION PHASE

The Formal Definition Phase consists of 3 sub-phases:

- An organizational sub-phase during which the Producer and the Archive have to define together the work organization and scheduling, the documents to be produced and the points which require more in-depth analysis.

- A formal definition sub-phase which is the most crucial part of the phase. It may be broken down into two parts:
    - ♦ A complete and precise definition of the Digital Objects to be transferred and the way they will be organised in the form of SIPs. This aspect will be formally specified in this recommendation.
    - ♦ A set of complementary points on contractual and legal aspects, the transfer schedule, validation conditions, etc. which will be covered by one or more appropriate documents.

- Another sub-phase during which the two parties will draw up and approve a ‘Submission Agreement’ defining the data to be transferred, the conditions for the transfer (for instance a specification of the Data Submission Session) and the validation.

2.3 THE TRANSFER PHASE

The Transfer Phase consists of two parts for:

- Implementing tests in order to validate the whole transfer chain.
- Performing the transfer itself. A key aspect of this part is the packaging of the Digital Objects in SIPs.

Packaging of the Digital Objects to be transformed in SIPs is also specified in this Recommendation in section 5 “Transfer Phase and Validation Phase Specification”.

2.4 THE VALIDATION PHASE

The Validation Phase consists of two parts for:

- Implementing tests in order to validate the systematic validation part.
- Performing the systematic and in-depth validation.

The purpose at this stage is to be able to check that the Digital Objects delivered are those which are expected and that they comply with what was defined during the Formal Definition Phase. In this document, only the systematic validation is taken into account.

Remark: two validation levels have been defined in the Formal Definition Phase (see [1]), systematic validation and in-depth validation. The systematic validation is the initial validation described here, carried out after each transferred SIP. The in-depth validation is not treated here –only cited in figure 5 as a future step-, and may be carried out when there is a coherent package of information, or at the end of the Producer-Archive Project when all the Data Objects are present.

3. GENERAL PRINCIPLES FOR IMPLEMENTING THE PRODUCER-ARCHIVE INTERFACE

3.1 OBJECTIVES

The major objectives may be summed up as follows:

During the Formal Definition Phase: develop a Plan for the Digital Objects to be transferred later (Plan of Objects to be Transferred – POT), which is sufficiently precise to meet the Producer's and the Archive's needs:

- For the Producer, 'sufficiently precise' means that he knows clearly which Digital Objects he will produce, that he knows how to do it and by which means.
- For the Archive, 'sufficiently precise' means that it knows that it will be capable of understanding the Submission Information Packages (SIPs) to be transferred to it so it can create the **Archival Information Packages** (AIPs) in compliance with the OAIS Model (possibly with additional Digital Objects from other Producers or other information already present at the Archive).

In addition, it should be possible:

- to use the POT with software for automation, checking and validation purposes,
- to represent the POT visually in a way that makes it easy for human beings to understand,
- to create such a POT for various other contexts, disciplines and organizations.

During the Transfer Phase: the means used should enable checking that:

- the schedule is respected,
- the procedures previously defined are applied (e.g. packaging),
- the operation runs well technically.

During the Validation Phase: the means used should enable checking that:

- any Digital Object transferred is an expected Digital Object in the previously defined POT,
- this Digital Object complies with the characteristics defined in the POT, including the number expected,
- the required structure of the data is preserved, such as hierarchical structure of files in directories,
- the Digital Objects have been transferred in the right order (sequencing constraints).

3.2. DATA OBJECTS AND TRANSFER OBJECTS

Data Objects, in the widest meaning of the word consist of one or more Digital Objects, in other words, one or more bit sequences (e.g., files), that are typically stored and used together.

A collection of Data Objects is a set of Data Objects having in common some properties. The use of collections will make it possible to organise all of the Data Objects to be delivered into easily understandable sets.

A Transfer Object is a structured and organized set of one or more Data Objects that are to be transferred to the Archive, whose characteristics are described by a Transfer Object Type. A Transfer Object Type is a leaf and granule of the POT.

However in this document, the phrase '**Transfer Object'** will be used to refer to an instance, while '**Transfer Object Type**' or 'type of Data Object' will be used to refer to the class of a Transfer Object.

3.3 PROCESS DESCRIPTION[1]

The PAIMAS (see [1]) describes the actions to be treated by the Archive and the Producer during the Formal Definition, Transfer and Validation Phases. The following process is based on the action tables and the chronology defined in [1]. Our main objective here is to define the process supporting development of the POT and creation of the SIPs[2].

The main steps of the global process are the following:
During the Formal Definition Phase:

- POT design.
- SIP design, definition of the grouping constraints and the sequencing constraints between SIPs.

During the Transfer and Validation Phases:

- SIP creation by the Producer.
- SIP transfer between the Producer and the Archive.
- SIP reception and validation by the Archive.
- Anomaly management by the Archive and the Producer.

During the Formal Definition Phase, the following takes place:

1. POT design[3]

   One of the objectives of the Formal Definition Phase is to define the POT. The POT provides a global view of the types of Objects to be transferred –the **Transfer Object Types**- and the relationships between them. The POT may be structured into collections.
   The POT is not an information organization Model within the Archive. It should simply enable the two partners to agree on the content of the information to be transferred and should facilitate the way this transfer is done. The POT is the basis for negotiating a Submission Agreement.

   In the POT, each Transfer Object Type is described in detail by a **Transfer Object Descriptor**. A collection of Transfer Object Types, or a collection of collections, is described by a **Collection Descriptor**. A Descriptor has a standardized structure derived from a Model called **Descriptor Model**.

   In the rest of the document, the term 'Descriptor' alone is used for a Transfer Object Descriptor or a Collection Descriptor.

[1] *See sections 4 and 5 for the implementation of the Formal Definition, Transfer and Validation Phases. See section 6 for the "Change management after completion of Submission Agreement" part.*

[2] *The sub-phases « Delivery schedule, Definition of Transfer Conditions, Validation Definition » of the Formal Definition Phase are not treated in this document.*

[3] *See section 3.4 "Design of the Plan of Objects to be Transferred" for a precise description of the POT design. The implementation is described in detail in section 4 "Formal Phase Specification".*

The POT design follows the chronological steps below (figure 1, UML Use Case representation):

- Descriptor Models creation[4]:
The Descriptor contains a set of information for describing a Transfer Object Type or a collection, and for defining its relations within the POT.
The Models of a Producer-Archive Project are derived from the generic standard Models described in this document or from domain Models.

- Descriptors creation[5]:
Each Digital Object category to be transferred is then described by a Descriptor. A Descriptor is created from the Models adapted to the project.

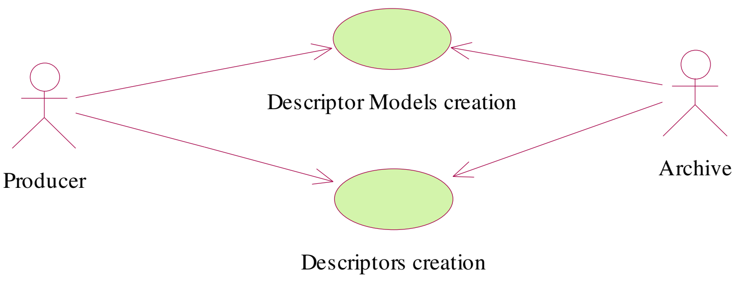


Figure 1: POT design

2. SIP design, definition of the grouping constraints and the sequencing constraints between SIPs[6]
(Figure 2 is a UML Use Case representation.)
The Data Objects are transferred between the Producer and the Archive in the form of SIPs. A SIP identifies the indispensable information for effective retrieval of bit sequences: these are object identifiers, reference to data containers, etc.

The Producer and the Archive define the project constraints: linked to the Producer (production planning, …), to the Archive (validation, …), to the number of Objects to be transferred, to the transfer (transfer capacity, …), or to the Application Process for each of the partners.

These constraints impact on the one hand the grouping of the Objects inside a SIP, on the other hand the sequencing constraints between the SIPs.

A SIP may contain several objects described by the same Descriptor or by different Descriptors, depending on the grouping rules defined by the Producer and the Archive (from the previous project constraints). These rules define the different SIP categories for the project. A SIP category is what is called a **SIP**

4 See section 3.4.4 for the development of a project Descriptor Models dictionary.

[5] See section 3.4.5"Descriptor creation" for the development of Descriptors describing what is to be delivered during the Producer-Archive Project..

[6] See sections 3.6.1 "SIP design", 3.6.3 "SIP grouping constraints definition" and section 3.6.4 "SIP sequencing constraints definition" for a detailed description of this step.

Type.

The sequencing constraints are the rules that apply between the SIPs to organize which ones must be delivered first, or before other ones.

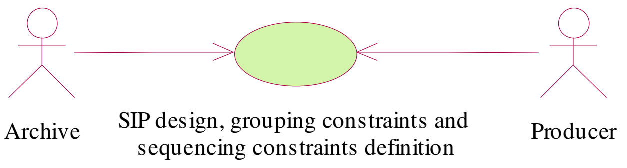

Archive
SIP design, grouping constraints and sequencing constraints definition
Producer

Figure 2: SIP design, grouping constraints and sequencing constraints definition

During the Formal Definition Phase (figures 1 and 2), all actions can be done by the Producer and/or the Archive
At this stage of negotiation between the Producer and the Archive, some characteristics of the objects to be delivered may not be known. How and when these unknowns will be defined should be detailed in the Submission Agreement. However, both the Archive and the Producer should each have the necessary information for defining and approving the POT.

Figure 3 shows the global process in a dynamic view dynamic using a UML activity diagram.

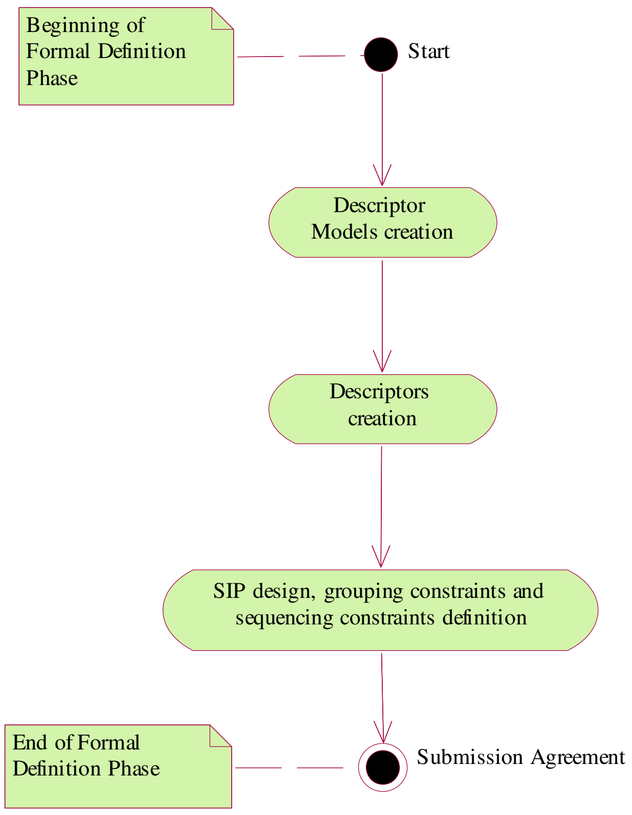

Beginning of Formal Definition Phase
Start
Descriptor Models creation
Descriptors creation
SIP design, grouping constraints and sequencing constraints definition
End of Formal Definition Phase
Submission Agreement

Figure 3: Formal Definition Phase general process

During the Transfer and Validation Phases, the following takes place[7]:
(Figure 4 is a UML Use Case representation, figure 5 shows the global process in a dynamic view using a UML activity diagram).

1. SIP creation by the Producer.
During the Transfer Phase, the SIPs will be created from a generic SIP Model, for the objects to be delivered, according to the grouping constraints defined during the Formal Definition Phase. This document recommends one generic SIP Model derived from a packaging standard[8].

2. SIP transfer between the Producer and the Archive.
This transfer is performed according to the Formal Definition Phase and following the sequencing constraints.

3. SIP reception and validation by the Archive.
After the reception of a SIP, the Archive performs the initial validations on the received SIPs to check that:

[7] *See section 5 "Transfer Phase and Validation Phase Specification" for the detailed following steps.*

[8] *See section 3.6.2 "Generic SIP Model".*

- The SIP conforms to the associated SIP Type.
- The SIP meets the sequencing constraints.
- The embedded Digital Objects conform to the associated Descriptor Model and the POT (content and number).
- The objects embedded in the SIP are those expected.

4. Anomaly management.
In case of detection of an anomaly, the Archive notifies the Producer.

Figure 4 shows that during the Transfer and Validation Phases, the roles of the Producer and the Archive are distinct for some actions.

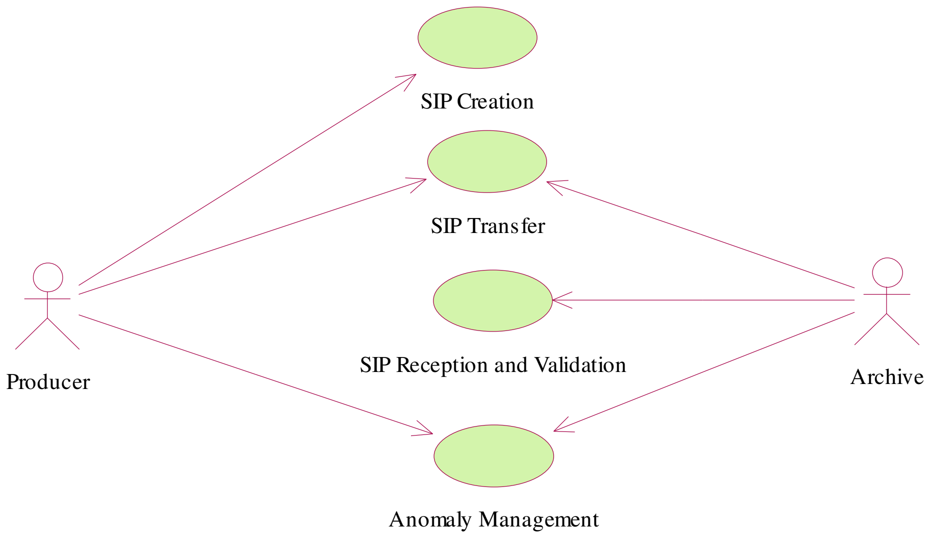


Figure 4: Transfer and Validation Phases

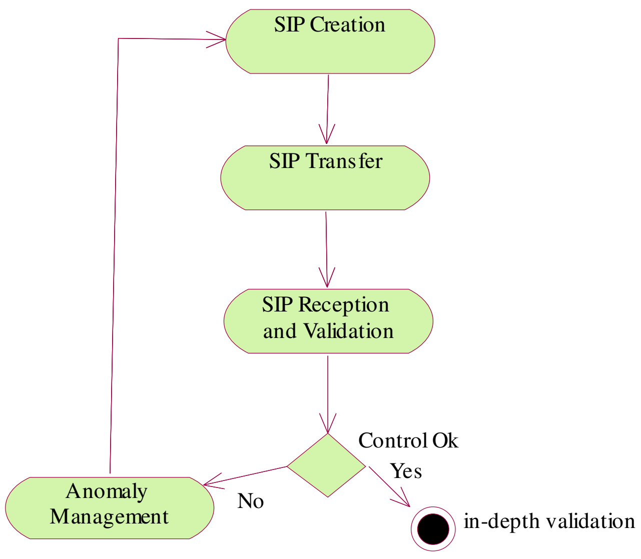


Figure 5: general process, from SIP creation to SIP validation

3.4 DESIGN OF THE PLAN OF OBJECTS TO BE TRANSFERRED (POT)

3.4.1 POT APPROACH

For a given Producer-Archive Project, the POT is developed during the Formal Definition Phase.

The POT must be sufficiently clear and precise for both partners to be able to make a formal commitment in the framework of the 'Producer-Archive Project'. The development of the POT assumes that the Producer and the Archive have done their iterative groundwork.

Example:

To describe a collection, it would for instance be suitable:

- *To define in writing what this collection would include.*
- *To give this collection a unique identifier and an explicit name.*
- *To specify what the parent collection is.*
- *To specify that the collection description will take the form of metadata in compliance with a specified metadata standard.*

But this does not, however, mean creating the metadata file, ahead of time, during the Formal Definition Phase.

To describe the objects in this collection, it would for instance be suitable to:

- *Give the essential information on the content of these objects: for instance, to indicate that the data contain the calibrated magnetic field vector expressed in a given inertial coordinate system;*

- *If necessary, specify the spatial or temporal resolution;*
- *Indicate whether the file format will comply with a specific standard format;*
- *Give average quantitative indications of the size of the files;*
- *Specify that a syntactical description will be created in EAST language or refer to a standardized data format*
- *Specify that a dictionary complying with the DEDSL Recommendation (see [B2] and [B3]) and describing the semantics of the fields will be made;*
- *Specify methods, constraints, or procedures that ensure objectives are met in the data transfer.*
- *Etc.*

This does not mean, however, creating the EAST description (see [B4]) and the semantic description during the Formal Definition Phase.

However, in another context, the Archive might require the complete semantic description before any Submission Agreement.

We can see through these examples that the boundary between supplying sufficiently precise and clear information in the POT and the supply of metadata during the Transfer Phase may vary and that both partners in the project will have to agree on the level of detail required.

The Modelling method proposed is thus very flexible and can be adapted to varying levels of detail required for given projects.

Moreover, in this Recommendation, we deliberately do not tie the description of the Transfer Object to the way in which this object is transferred. A given SIP Type may in fact be applied to several object categories.

The POT gives a complete and overall view of all Digital Objects to be transferred as part of the Producer-Archive Project. These objects are described by Transfer Object Descriptors. By Transfer Object Descriptor, we mean the supply of a set of information which characterizes the described Objects with the required level of details. The content of a Descriptor may vary greatly according to the projects and the information available during the negotiation.
The Plan will thus be structured into Descriptors (Transfer Objects Descriptors and Collection Descriptors).
These Descriptors are built from Descriptor Models adapted for the project. These project specific Models are themselves derived from domain specific Models and /or the generic specific Models defined in this recommendation. This is why each Producer-Archive Project should create specialized versions of the proposed generic Models by adding, eliminating or modifying the attributes of these Models.

This is a fairly simplified view because when developing Descriptors, it is always possible, depending on the needs, to modify, complete or specialize Model dictionaries.

Important remark: the leaves of the POT are the granules[9] of the plan (the description of a Data Object and its Metadata can be grouped in the same Transfer Object Descriptor: that means that this set represents a granule of information. This set can't be separated in the POT and later in the SIP.

[9] *The granules of the POT are elementary entities that can not be split and are described by the Transfer Object Types.*

3.4.2 POT CONSTRAINT

All the Objects defined in a Descriptor must be delivered together in the same SIP. However, a single SIP may contain several Transfer Objects described by different Transfer Object Descriptors.
In the case where the Transfer Object described by the Descriptor has many occurrences, the **same SIP** must contain all the Digital Objects associated with at least **one occurrence**.
However this constraint will greatly simplify the management of the progress of the actual delivery of Objects by comparing to what is described in the POT.

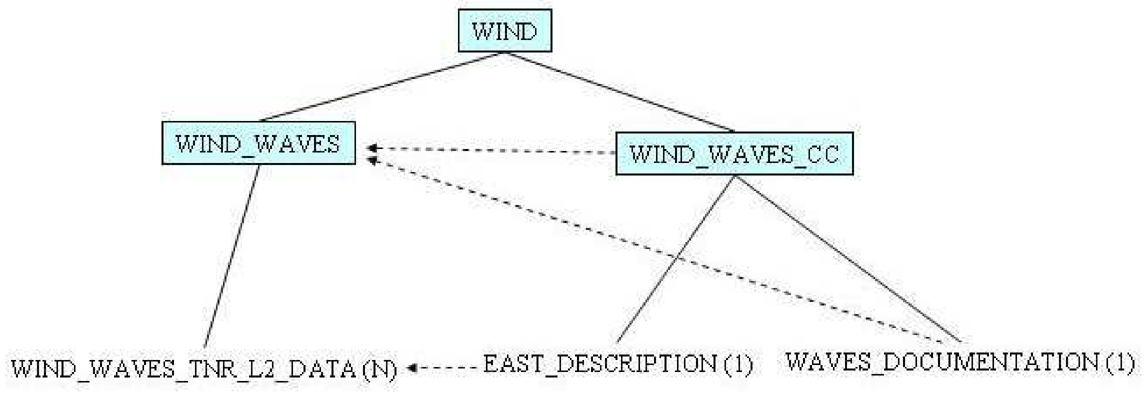
3.4.3 POT VISUALISATION

The POT should enable the Producer and the Archive to have a clear view and understanding of the objects to be transferred. This Plan should also elementary entities that can not be split enable them to use automatic processing and visual displays for monitoring the project and validating deliveries, whereas the implementation of the Plan in a computer language does not allow for visual displays which are easily understandable to the human eye. However, it may be possible to have tools that automatically convert to a displayable form.

The POT as previously defined is a tree structure to which all of the Descriptors are linked (there are no orphan elements). This tree structure consists of descending hierarchical relations between collections of collections, collections of Transfer Object Types and Transfer Object Types, as well as transversal relations between objects. Finally, each node of the tree structure, whether this involves collections or Transfer Object Types is defined by information contained in its corresponding Descriptor.

The POT may thus be represented with the following levels of detail:
- Hierarchical tree structure representation.
- Hierarchical tree structure representation and transversal relations.
- Complete description of each Plan's node (i.e., the Descriptors).

Moreover, it is useful that the graphic representation of the POT is used to distinguish between the different object categories and the number of times each object occurs, as it is the case in the example shown in figure 6 below.



- *The WIND_WAVES node is a collection of sets of Data Objects (e.g. TNR L2 data), while the WIND_WAVES_CC node gathers metadata for the WAVES experiment and for the TNR L2 data.*

- *WIND_WAVES_TNR_L2_DATA is the Transfer Object Type for the TNR L2 data and characterises these data to be transferred.*
- *EAST_DESCRIPTION is the Transfer Object Type for the syntactic description of the TNR L2 data. The WAVES_DOCUMENTATION characterises the description of the WAVES experiment.*
- *The WIND_WAVES_TNR_L2_DATA, EAST_DESCRIPTION and WAVES_DOCUMENTATION nodes are leaves of the tree –the Transfer Object Types- (in parentheses is the number of objects to be transferred).*
- *The dotted lines show the transversal relations[10].*

Figure 6: Example of hierarchical tree structure representation

3.4.4 DICTIONARY FOR PROJECT DESCRIPTOR MODELS

3.4.4.1 Process for the Dictionary development

The project Descriptor Models are negotiated by the Archive and the Producer during the Formal Definition Phase.
Figure 7 shows the possible process for the dictionary development, and for the specialization of a generic Descriptor Model into a Producer-Archive Project Descriptor Model[11] .

A Transfer Object Descriptor can describe any kind of Transfer Object, a Collection Descriptor can describe any kind of collections as defined in the Producer-Archive Project terminology. The Producer-Archive Project Descriptors will be built from Producer-Archive Project Descriptor Models. Producer-Archive Project Descriptor Models are derived directly from the generic Descriptor Models or from a Domain Descriptor Model specialized for a domain. These Domain Descriptor Models are themselves built from the standardized generic Descriptor Models or from another Domain Descriptor Model.

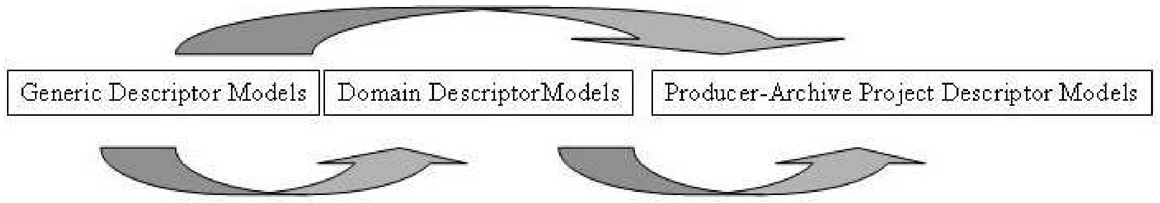

Generic Descriptor Models
Domain DescriptorModels
Producer-Archive Project Descriptor Models

Figure 7: Process for the development of the Project Descriptor Models

This recommendation proposes two generic Descriptor Models (a generic Transfer Object Descriptor and a generic Collection Descriptor), including the different information categories, which can easily be specialized by the suppression of optional attributes or addition of attributes.

The following sections describe:

- The structure of the generic Descriptor Models.
- The specialization phase.

[10] See "association" in section 3.4.4.2 "Generic Descriptor Model definition.
[11] See also section 4 in [1].

3.4.4.2 Generic Descriptor Models definition

We are not referring at this stage to a particular implementation choice[12]. We are first defining an abstract view of the generic Descriptor Models (Models for the Transfer Object Descriptor, and for the Collection Descriptor).

3.4.4.2.1 Transfer Object Descriptor

The generic Transfer Object Descriptor is expected to be complete enough to provide the information needed for the description of different types of objects to be transferred to the archive and it is expected to be flexible enough to adapt to the specializations required by each Producer-Archive Project.

The Transfer Object Descriptor is made up of a set of attributes and values which are assigned to these attributes in order to characterize the described Digital Objects. These attributes are divided into the following sections (the sections “Relationships” and “Object content” are made up of sub-sections):

- **Descriptor identification**.
- **Transfer Object description**: set of attributes giving all the information needed to describe the Transfer Object, such as the title, the content, and the size.
- **Relationships within the POT**: aggregation relationship of this Descriptor inside the POT, and directional links to other nodes (Descriptors) of the POT.
- **Transfer Object content**: describes all the Data/Metadata Objects included in this Descriptor, even complex Data Objects made up of several parts. Each part is identified by an ID. Thus, the packaging will be easier and the validation will be possible. The “Transfer Object content” section can be repeated if necessary.

- **User Define Attributes**: user’s may incorporate additional attributes as needed to further specialize the Descriptors (also possible in each of the preceding sections).

The sub-sections are composed of the attributes described below. Each attribute is associated with a status (M for Mandatory, O for Optional), and an occurrence. The attributes themselves belong to groups that can be mandatory or optional. Inside an optional group, an attribute may be mandatory (on the contrary, inside a mandatory group, an attribute may be optional).

- **Identification (M, 1..1)**

  - **descriptor_model_ID (M, 1..1)**: identifies the Descriptor Model upon which this Descriptor is based. It may be the Descriptor Model as given in the standard or it may be a specialized version.

  - **descriptor_ID (M, 1..1)**: Descriptor Identifier, used to identify the nodes in the POT. This ID is used in the associated SIP Model ID to refer to the Descriptor.

  - **version (M, 1..1)** : version of the Descriptor Model. This allows tracking updates to the identified Descriptor Model.

- **Transfer Object Description (M, 1..1)**

  - **title (M, 1..1)**: extensive descriptive phrase used as name of the Transfer Object Type.

[12] *See section 4.2 “POT specification”.*

  - **description (M, 1..1)**: explanatory text describing the meaning of the Transfer Object Type.

  - **transfer_object_occurrence** (**M, 1..1**): number of Transfer Objects of this Transfer Object Type (i.e., this descriptor_ID) to be transferred. This may be one or more or an interval. This number may not be known at the time of descriptor creation and instantiation. In the case of a unique value, then min_occurrence = max_occurrence = value.

    - **min_occurrence:** integer value, or 'unknown' (<max_occurrence).

    - **max_occurrence:** integer value, or 'unknown' (>min_occurrence).

  - **transfer_object_size (O, 0..1)**: estimated average size, including units, of a Transfer Object of this type. This size may not be known at the time of descriptor creation and instantiation.

- **Relationships within the POT (M, 1..1)**

  - **parent_collection (M, 1..1)**: identifier of a Collection Descriptor that provides an aggregation view of the object types under this Descriptor. The highest level Descriptor must refer to the name of the Archive Project, the value for its parent_collection is 'none'.

  - **association (O, 0..N)**: used to describe another relationship between the Transfer Object Type of this Descriptor and other object types of the Producer-Archive Project (Transfer Object Types, types of Data Objects inside Transfer Object Types, collections).

    - **target_ID (M, 1..1) :** identifier for an object type (s) of the Producer-Archive Project to which a relationship from this Transfer Object Type is established.

    - **relation_description (M, 1..N):** description of the relationship from this Descriptor's object type(s) to the target object type(s)**.**

      - **relation_type (M, 1..1):** type of relation – for example "DED, Syntax, Context, Provenance, Reference, Fixity - between the Transfer Object type described by this Descriptor and the object type(s) identified by target_ID. This is a directional relation (from descriptor_ID towards target_ID).

      - **relation_textual_description (O, 0..1)**: free-text description of the relation.

- **data_object content (M, 1..N)**. A Transfer Object Type may be composed of multiple Data Object Types, and a Data Object Type may include other Data_Object Types thereby forming a hierarchical structure.

  - **data_object_ID (M, 1..1)**: an identifier of the data object type. Needed if the Transfer Object Type is composed of several different separated data object types. A complete Transfer Object Type instance (i.e., Transfer Object) must be delivered to the archive at the same time to facilitate validation and thus may influence the definition of Transfer Object Types.

  - **data_object_occurrence (M, 1..1)**: number of data objects of this type to be delivered under the definition of this Transfer Object Type. This may be one or more or an interval. This number may not be known at the time of descriptor creation and instantiation. In the case of a unique value, then min_occurrence = max_occurrence = value.

    - **min_occurrence:** integer value, or ‘unknown’ (<max_occurrence).
    - **max_occurrence:** integer value, or ‘unknown’ (>min_occurrence).

- o **data object_description (O, 0..1)**: Explanatory text describing the meaning of this data object type.
- o **data_object_association (O, 0..N)**: used to describe the relationship between the Data Object type under consideration and the other object types of the Producer-Archive Project (Data Object types, Transfer Object Types, collections). These are transversal links specifying the type of relation.
    - **target_ID (M, 1..1) :** identifier for an object type of the Producer-Archive Project to which a relationship from this Data Object type is established.
    - **relation_description (M, 1..N):** description of the relationship from this Data Object type to the target object type(s).
        - **relation_type (M, 1..1):** type of relation –for example “DED, Syntax, Context, Provenance, Reference, Fixity- between this Data Object type and the object type identified by target_ID. This is a directional relation (from data_object_ID towards target_ID).
        - **relation_textual_description (0, 0..1)**: free-text description of the relation.
- o **data object_format (O, 0..1)**: identification of format of this Data Object type in the Transfer Object type (PNG, PDF, CDF, Flat Binary, Flat ASCII, …)
    - **mime_type**: one of the type existing in this list. In the case of a different one, then,
    - **other_format**: made up of **format_name** and **format_description**, texts to identify the format.
- o **data_object_content (O, 0..N)**. When the Transfer Object Type consists of a hierarchical structure of Data Object types, this section is repeated to describe each of the nested types.
    - **Etc**.

3.4.4.2.2 Collection Descriptor

The Collection Descriptor is defined to support one or more collection views of the objects to be transferred (in this case there is no object attached to a Collection Descriptor, and so no Data Object type description).

- **Descriptor identification**.
- **Collection description**: set of attributes giving all the information needed to describe the collection, such as the title, the content, and the size.

- **Relationships within the POT**: aggregation relationship of this Descriptor inside the POT, and other directional relationships to other objects of the POT.
- **User Define Attributes**: user's may incorporate additional attributes as needed to further specialize the Descriptors (also possible in each of the preceding sections).

The sub-sections are composed of the attributes described below. Each attribute is associated with a status (M for Mandatory, O for Optional), and an occurrence. The attributes themselves belong to groups that can be mandatory or optional. Inside an optional group, an attribute may be mandatory (on the contrary, inside a mandatory group, an attribute may be optional).

- **Identification (M, 1..1)**
  - **descriptor_model (M, 1..1)**: identifies the Descriptor Model upon which this Descriptor is based. It may be the Descriptor Model as given in the standard or it may be a specialized version..
  - **descriptor_ID (M, 1..1)**: Descriptor Identifier, used to identify the nodes in the POT. This ID is used in the associated SIP Model ID to refer to the Descriptor.
  - **version (M, 1..1)**: version of the Descriptor Model. This allows tracking updates to the identified Descriptor Model.
- **Collection Description (M, 1..1)**
  - **title (M, 1..1)**: extensive descriptive phrase used as name of the collection.
  - **description (M, 1..1)**: explanatory text describing the meaning of the collection.
  - **collection_size (O, 0..1)**: Approximate size of the collection, including unit.
- **Relationships within the POT (M, 1..1)**
  - **parent_collection (M, 1..1)**: identifier of a collection Descriptor that provides an aggregation view of the object types under this Descriptor. The highest level Descriptor must refer to the name of the Archive Project, the value for its parent_collection is 'none'.
  - **association (O, 0..N)**: used to describe the relationship between this collection and other object types of the Producer-Archive Project (Data Object types, Transfer Object Types, collections).
    - **target_ID (M, 1..1) :** identifier for an object type of the Producer-Archive Project to which a relationship from this collection is established.
    - **relation_description (M, 1..N):** description of the relationship from this collection's Transfer Object Type(s) to the target object type(s).
      - **relation_type (M, 1..1):** type of relation – for example "DED, Syntax, Context, Provenance, Reference, Fixity - between the object type(s) described by this Descriptor and the object type(s) identified by target_ID (directional relation).

- **relation_textual_description (O, 0..1):** free-text description of the relation.

3.4.4.3 Specialization of the generic Descriptor Models

The previous generic Models (Transfer Object Descriptor Model and Collection Descriptor Model) are intended to cover the different possible situations and contains a small set of mandatory attributes, some additional optional attributes, a "non defined" attribute and a "non-defined" section.

These generic Descriptor Models may be specialized to achieve the applicable level of description desired by the Producer-Archive Project:

- Specialization of a domain Model from the generic Model[13].
- Domain specialization for creating a project Model (other intermediary levels may be considered if necessary). Project Descriptor Models may be created directly from the generic Descriptor Models.

To do this, the standard generic Models may undergo the following modifications:

- Addition of an attribute by using the:
  - "Non definite section" (exists in each part of the Descriptor): made up of a unique attribute, or a set of attributes.
- Elimination of an optional attribute.
- Modification of existing attributes: by occurrence restriction, by modification of status attribute (e.g. optional changed in mandatory), by text restriction (use of list). The name of an attribute should not be modified (to adapt to the project terminology for example), in order that the tools manipulating Descriptors remain usable.

3.4.5 DESCRIPTOR CREATION

The creation of Descriptors is definitely the most crucial task during the Formal Definition Phase since it is at this stage that we clearly define the Objects to be transferred. This creation allows for the POT to be built.

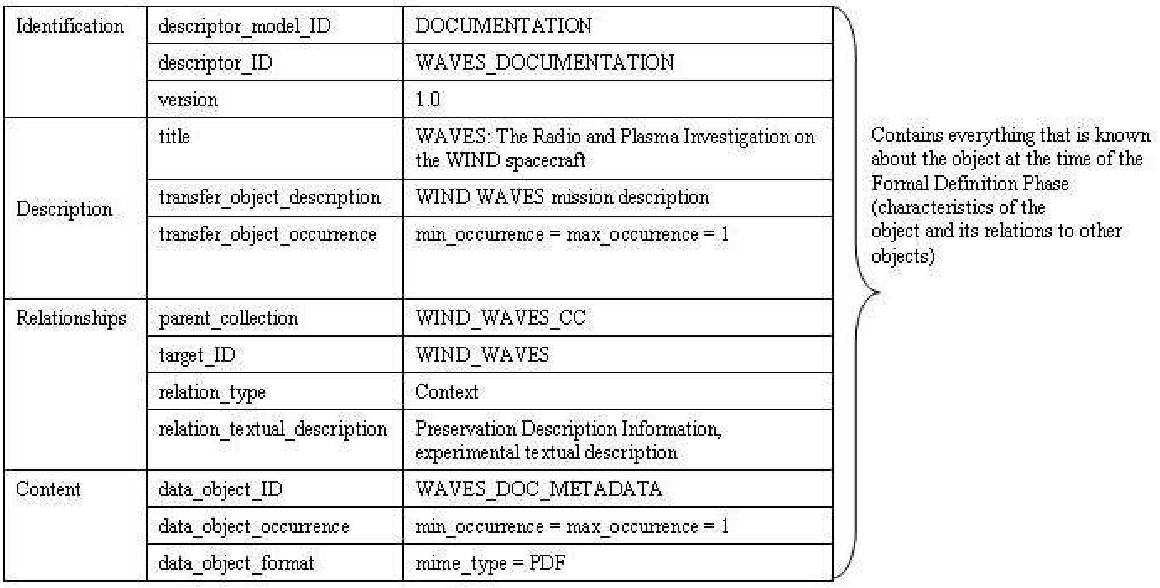
This task will consist for example of:

- Identifying all of the project data collections.
- Organising these collections in a hierarchical tree structure.
- Defining a Collection Descriptor for each of the collections.
- Defining a Transfer Object Descriptor for the objects in each collection.

Descriptors can be instantiated by different persons and in several times during the Formal Definition Phase (all information are not necessarily available at the same time).

[13] *See section 4 of [1].*

Figure 8 shows an example of a Transfer Object Descriptor for a description document for space physics (see also figure for the tree).



Identification	descriptor_model_ID	DOCUMENTATION
	descriptor_ID	WAVES_DOCUMENTATION
	version	1.0
Description	title	WAVES: The Radio and Plasma Investigation on the WIND spacecraft
	transfer_object_description	WIND WAVES mission description
	transfer_object_occurrence	min_occurrence = max_occurrence = 1
Relationships	parent_collection	WIND_WAVES_CC
	target_ID	WIND_WAVES
	relation_type	Context
	relation_textual_description	Preservation Description Information, experimental textual description
Content	data_object_ID	WAVES_DOC_METADATA
	data_object_occurrence	min_occurrence = max_occurrence = 1
	data_object_format	mime_type = PDF

Contains everything that is known about the object at the time of the Formal Definition Phase (characteristics of the object and its relations to other objects)

Figure 8: Example of a Transfer Object Descriptor

3.5 LINK BETWEEN DESCRIPTOR AND SIP

Reminder:

- All the objects defined in a Transfer Object Descriptor must be delivered together in the same SIP.

Figure 9 shows the links between a Transfer Object Descriptor and a SIP (there is no link between a Collection Descriptor and a SIP, as there is no described object to transfer in this case):

- A Transfer Object Descriptor describes an object to be transferred (for example DA and DB describe two different and separate objects).
- An object is transferred inside a SIP.
- A SIP may contain several different objects. Depending on the project SIP grouping rules, different types of SIPs may be defined -see the following section- (for example the two objects associated with DA and DB may be transferred via a SIP of type sipS).
- A single SIP may contain several objects of the same Descriptor (for example a SIP of type sipS may contain several objects of DA, and exactly 2 objects of DB).
- A created SIP refers to its type (for example sipS),needed to make the link with the sequencing rules, and to the Descriptor of each embedded object (for example DA and DB), needed to make the link with the POT. This information is used after the transfer, for the validation.

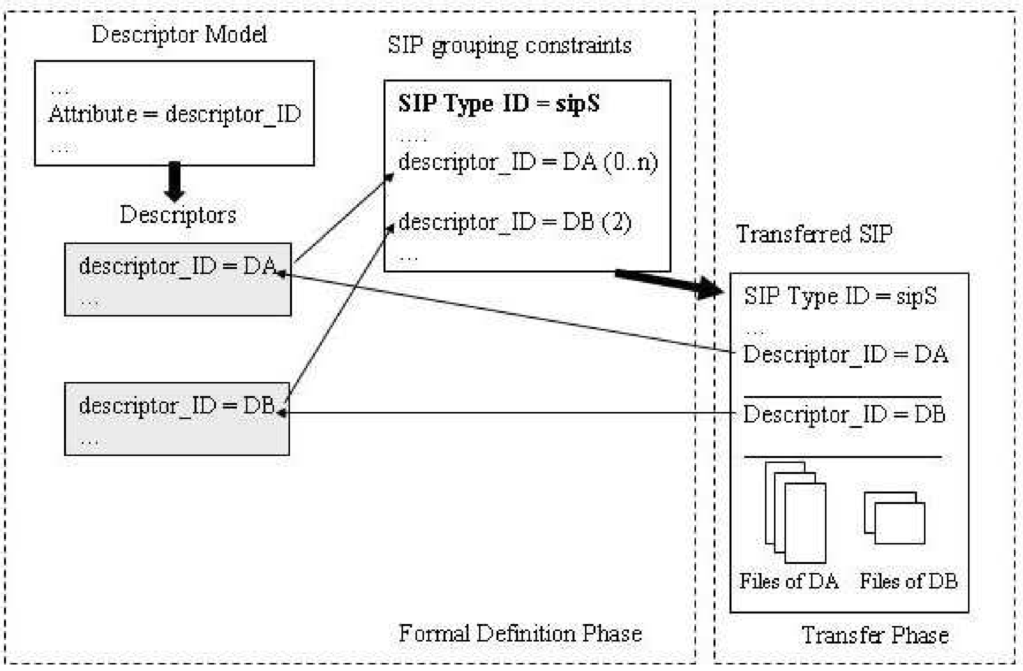


Figure 9: links between Transfer Object Descriptor and SIP

3.6 SIP AND CONSTRAINTS

3.6.1 SIP DESIGN

A SIP can simply be a coherent group of Data Objects which is transferred in the same package. A SIP may also contain a pointer to the Data Objects location in case the Data Objects are not physically transferred in the SIP. For example, a URL or FTP site may be referenced in the SIP.

These Data Objects may be grouped together in a SIP for very different reasons. For example:

- Due to Archive procedures, it may be necessary for the Archive to have several Data Objects made available within the same SIP package in order to be able to create the corresponding AIPs.
- It might be useful to optimise the transfer: if the Data Objects are very small, a decision may be taken to group several of these Data Objects together in a single package so as not to have too many packages.

We shall consider each Data Object as a whole, which means in particular that the different bit sequences that make up the Data Object may not be separated into elements transferred in separate SIPs. However, a single SIP may be broken into several packages.

Figure 10 shows the process to build the SIPs:

- A SIP is created from the generic SIP Model and the table of SIP grouping constraints.
- The table of SIP sequencing constraint is used to transfer the SIPs in the right order.

The following sections describe the constraints tables and the generic SIP Model.

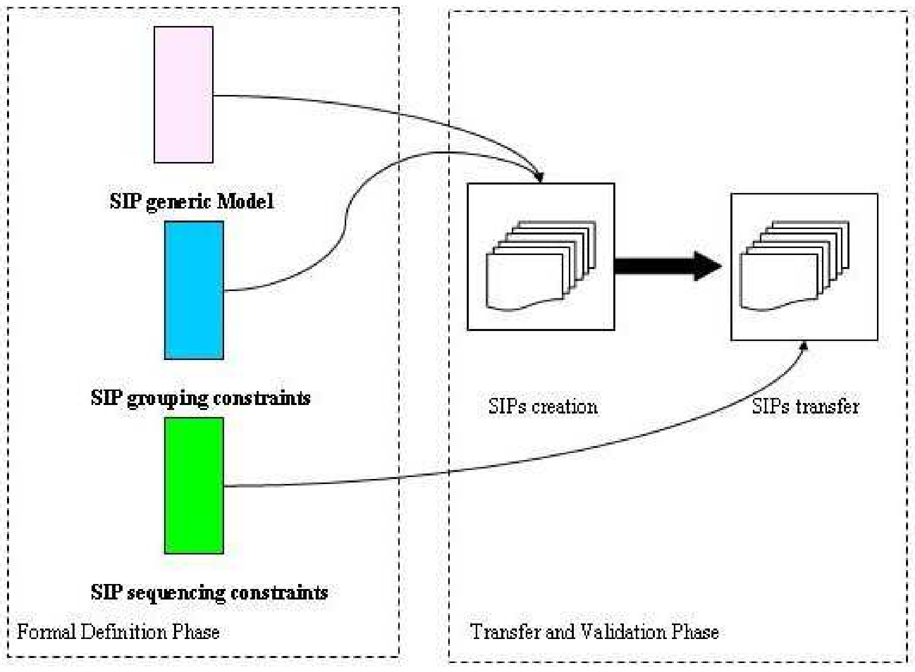


Figure 10: SIP and constraints

3.6.2 GENERIC SIP MODEL

See section 4.3 for the implementation part.

The SIPs will be built from a generic SIP Model.
The SIP is based on a packaging standard (see figure 12). This packaging standard is specialized with a certain number of attributes directly linked to the PAIS logic to create the generic SIP Model. It contains the attributes described here below.
The Generic SIP Model can be specialized to create Domain or Producer-Archive Project SIP Models.

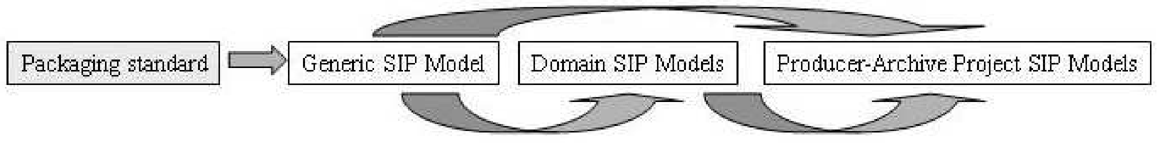


Figure 12: Generic SIP Model creation

A SIP can contain one or different categories of Transfer Objects. A SIP Type characterizes the content of a SIP, that means the Tranfer Object Types it can include.

The Archive must analyze each SIP content and perform the initial validations. Each SIP should contain the information required for this analysis and for these validations (and to automate the process):

- Global SIP information.
- Information associated with each embedded Transfer Object.

Each attribute is associated with a status (M for Mandatory, O for Optional), and an occurrence (see figure 13):

- **Global information (M, 1..1)**:
    - **SIP ID (M, 1..1)**: identifier of the delivered SIP.
    - **Producer ID (M, 1..1):** this identifier (related to the Producer name and address) should enable the Archive to identify the origin of the SIP -the Producer, and should enable the Archive to send an acknowledgement.
    - **Project ID (O, 0..1)**.
    - **SIP Type ID (O, 0..1)** and **SIP Type location (O, 0..1)**: attributes that allow for the definition of the different SIP categories within a Producer-Archive Project. They define the SIP category to which the SIP under consideration belongs.
    - **SIP sequence number (O, 0..1)**: number indicating in which order the SIPs transferred are organized and have been sent. This number is unique. It can be used by the Producer to indicate the order, and by the Archive to validate the received SIPs. Also useful in the transfer of a collection of objects to check the order of arrivals (in particular for the last object).
    - **Sub sequence number for a SIP split into several packages**: this is used to identify the different parts of information of one SIP in the case this SIP has been split into several packages for the transfer, and to be able to rebuild it. This attribute is made up of:
        - **sequence position (O, 0..1)** (position of the package in the sequence of packages), and

    - **sequence size (O, 0..1)** (total number of linked packages).
- **Information associated with each embedded Transfer Object** (**M, 1..N**): all the objects included in the same Transfer Object must be gathered in a subset of the SIP.
    - **Descriptor_ID** (**M, 1..1**) and **Descriptor location** (**M , 1..1**) : object Descriptor identifier and file name and location: required for validation tests (each embedded object must comply with the POT content).
    - Map of links between Descriptor ID and delivered files names (data_object_ID).
    - **Last object** (**O, 0..1**): indicator specifying that it is the last object (among the objects, or collections corresponding to the same POT node). This attribute is useful if the number of objects in a collection is not known ahead of time and is not defined in the POT. When it is used, a sequence management rule has to be specified for transferring the objects (case in which, during transfer, the last object would not arrive last).
    - **Updated object** (**O, 0..1**): in case of a transferred object which has already been archived (same ID), due to the detection of an error in an archived object.
- **Other information** (**M, 1..1**)**:** this list is not exhaustive:
    - Way of retrieving the object (e.g. in the case of tar or zip files inside the SIP).
    - Way of reconstructing the object (e.g. in the case of an object split in several SIPs: number of SIPs, pointer towards the following SIP, « nil » pointer for the last, name of a reconstruction software if necessary).
- **Digital Object** (**O, 0..N**): delivered files. In certain cases, the object is not physically delivered (e.g. files already archived in a different site). In that case, the SIP will contain a link towards the object.

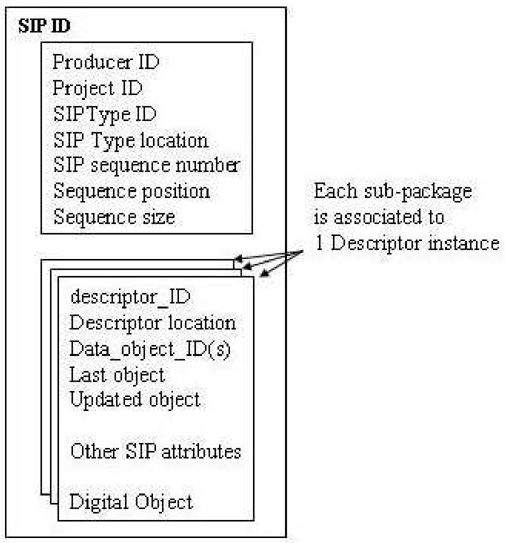


Figure 13: SIP structure

Apart from specific procedures to be used for renewing transfer of an object which has already been transferred (for instance following observation of an anomaly in the initial object), we have assumed that an object identified in the POT is transferred only once. The Producer and Archive thus have to ensure that the SIP definitions are coherent from this point of view.

3.6.3 SIP GROUPING CONSTRAINTS DEFINITION

Each Producer-Archive Project can define the SIP Types, that is the object categories that can be delivered together in the same package.
These SIP Types are described in a SIP grouping constraints table. This table of grouping constraints is defined by a list of the following (see section 4.4 for the implementation part):

- **SIP Type ID (M, 1..1)**: SIP Type identifier. It is referenced in each created SIP.
- A list of the authorized categories of objects, each category being associated to a Transfer Object Descriptor (identified by a Descriptor_ID). "Authorized" means that an actually transferred SIP of the type "SIP Type ID", may or not contain an object of Descriptor_ID type. However it can not contain an object associated with a Descriptor_ID that is not on this list. It is understood that a SIP must contain at least one object to be delivered.

    - **descriptor_ID (M, 1..1)**
    - **occurrence (O, 0..N)**: number of times the Transfer Object Descriptor may be present in the SIP. In the case of a collection of objects, the project can authorize the delivery in the same SIP of one or several objects associated with the same Transfer Object Descriptor. This value can be "0..N" if the number of objects in the same category, and contained in the same SIP, is variable. This occurrence must be inferior or equal to the attribute « transfer_object_occurrence » of the Descriptor under consideration.

3.6.4 SIP SEQUENCING CONSTRAINTS DEFINITION

In the transfer process, the sequencing constraints apply at the SIP level, which means on the transferred packages.

If there is no sequencing constraint, the SIPs may be transferred independently of each other in any order.
It may be necessary to specify in the Submission Agreement that a given Digital Object must be transmitted before or after another one.

For instance, in the case of a collection of files of scientific data all having the same syntactical structure described with EAST language, the archive might want to systematically verify the conformance of each file in the collection in relation to the EAST description of the files in this collection. In this case, the EAST description should be sent before the data files.

This constraint must be converted into a constraint on the different SIPs transporting these objects.

The sequencing constraints for a Producer-Archive Project should remain simple.
This is why we shall only define here a limited number of possibilities for expressing sequencing constraints.

The constraints that exist between two objects A and B may be totally independent of the constraints that exist between two other objects C and D. This has led us to define the concept of **constraints group** as being the set of objects related to each other by a set of dependent sequencing constraints.
The example referred to above concerning an EAST syntactical descriptor can in practice apply to several collections of distinct Data Objects, each having its own EAST description and belonging to the same Producer-Archive Project. We may thus define several independent constraint groups:

Example:
Constraint group Group_1:
The EAST description of the files in the A collection must be delivered before the files in this collection.
Constraint group Group_2:
The EAST description of the files in the B collection must be delivered before the files in this collection.

These constraints between SIPs are expressed during the Formal Definition Phase in a "SIP sequencing constraint table", using the following attributes:

Attribute name	Meaning	Value syntax	Occurrence
time_constraint_group	define the groups	identifier	0..1
serial_number_in_constraint	define the constraints within the same group	integer	0..1

- The use of these two attributes is optional.
- If "time_constraint_group" is defined in the "SIP sequencing constraint table", then "serial_number_in_constraint" is mandatory for this SIP.
- If there is no time constraint, they are omitted.
- If there are time constraints, but a unique group, time_constraint_group may be omitted.

If two SIPs SIP1 and SIP2 belong to the same group with:
serial_number_in_constraint = 1 for SIP1 and
serial_number_in_constraint = 2 for SIP2,
this means that the Digital Object(s) corresponding to SIP1 must be transferred before the Digital Object(s) corresponding to SIP2.
If three SIPs SIP1, SIP2 and SIP3 belong to the same group with:
serial_number_in_constraint = 1 for SIP1,
serial_number_in_constraint = 2 for SIP2 and
serial_number_in_constraint = 2 for SIP3,
this means that the Digital Object(s) corresponding to SIP1 must be transferred before the Digital Object(s) corresponding to SIP1, SIP2 and SIP3 and that there is no constraint between the Digital Object(s) of SIP2 and SIP3.

Example:
This example defines two groups of sequencing constraints which are independent of each other.
For each group, it specifies:
- *that the EAST Descriptor must be transferred first,*
- *that the objects corresponding to the collection descriptions (metadata describing the collection) and the Data Objects for these collections can then be transferred in any order for each collection.*

	SIP constraint description					
Sip Type ID	SIP1	SIP2	SIP3	SIP4	SIP5	SIP6
SIP content	Collection_1	Collection_2	Data_object_collection_1	Data_object_collection_2	EAST_descriptor_collection_1	EAST_descriptor_collection_2

time_constraint_group	Group_1		Group_1	Group_2	Groupe_1	Groupe_2
serial_number_in_constraint	2		2	2	1	1

Figure 11: example of sequencing constraints

Remark: it is not possible to impose constraints between specific objects belonging to 2 different groups each associated to a Descriptor. The sequencing constraints may apply between the 2 groups (and not between elements inside the same group or different group). Furthermore, in the case of constraints between 2 collections C1 and C2, if C1 must be delivered before C2, that means that the first object of C2 may be delivered only after the last delivery of C1.

4 FORMAL PHASE SPECIFICATION

In this section, we propose a particular implementation of the concepts described previously.
The proposed implementation is based on XML Schema and XML files.

4.1 GLOBAL VIEW

Figure 14 shows the main steps and the links between them. The attributes cited here are detailed in the following sections.

All the information coming from the Descriptors and the sequencing constraints populate an Ingest Base. This information is used on the one hand by the functions creating the POT, on the other hand by the functions managing and validating the transfer (e.g. transfer_object_occurrence, sip_type_ID).

This Ingest Base contains as well other information not cited here and necessary for the ingest process (contact information of Producer -Producer_ID-, …).

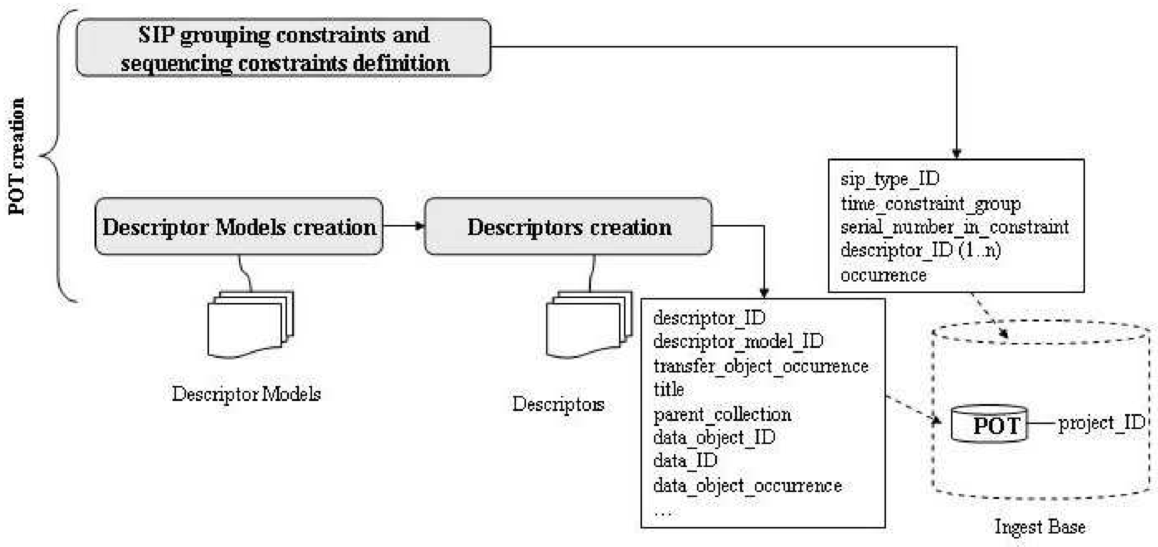


Figure 14: global view of the Formal Definition Phase Specification

4.2 POT SPECIFICATION

The following sections describe the steps for the POT creation. The generic Descriptor Models in 4.2.1 are part of the recommendation and are not to be done by the Producer-Archive Project, the given XML schemas are the input for the next step in 4.2.2.

4.2.1 GENERIC DESCRIPTOR MODELS IMPLEMENTATION

4.2.1.1 Generic Transfer Object Descriptor Model

Figure 15 shows the generic Transfer Object Descriptor Model. It is implemented with an XML Schema (see Annex A.1 for the XML source code).

This schema contains the attributes specified in section 3.4.4.2.1 “Transfer Object Descriptor”.
It is based on a hierarchical structure identifying the main descriptive parts of the object. The schema itself contains comments on each field. It is divided in 5 parts (that can be divided in their turn) « identification, content_description, relation, content, any,»:

- **identification**: cf. « Identification » section 3.4.4.2.1.
- **description**: cf. “Transfer Object description” section 3.4.4.2**.1**.
- **relation**: cf. “Relationships within the POT” section 3.4.4.2.1.
- **content**: cf. “Transfer Object content” section 3.4.4.2.1.
- **any**: cf . « User Define Attributes » section 3.4.4.2.1. This element may contain itself a sequence of elements (or another sequence) depending on the Producer-Archive Project needs.

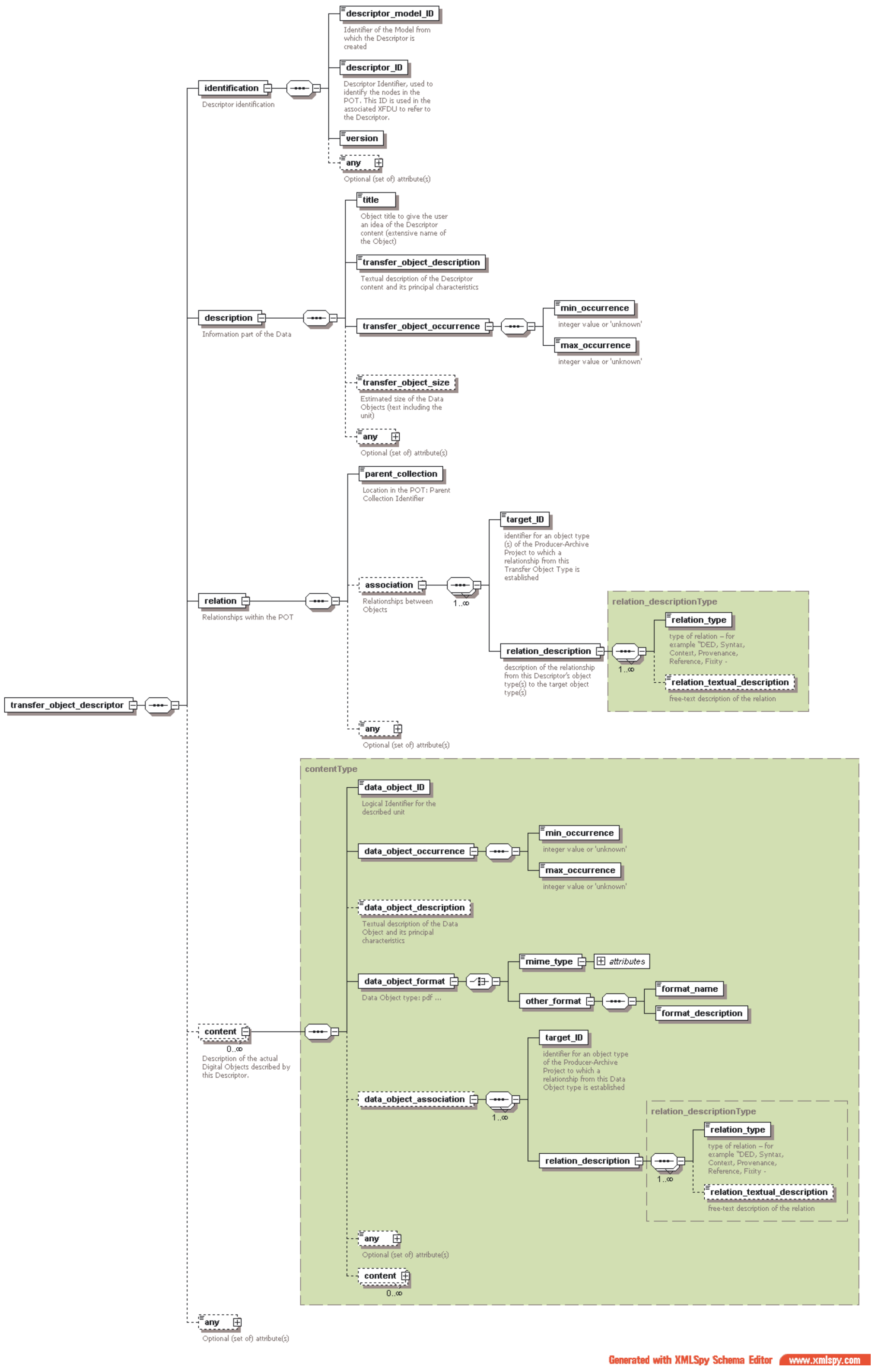

transfer_object_descriptor
identification
Descriptor identification
descriptor_model_ID
Identifier of the Model from which the Descriptor is created
descriptor_ID
Descriptor Identifier, used to identify the nodes in the POT. This ID is used in the associated XFDU to refer to the Descriptor.
version
any
Optional (set of) attribute(s)
description
Information part of the Data
title
Object title to give the user an idea of the Descriptor content (extensive name of the Object)
transfer_object_description
Textual description of the Descriptor content and its principal characteristics
transfer_object_occurrence
min_occurrence
integer value or 'unknown'
max_occurrence
integer value or 'unknown'
transfer_object_size
Estimated size of the Data Objects (text including the unit)
any
Optional (set of) attribute(s)
relation
Relationships within the POT
parent_collection
Location in the POT: Parent Collection Identifier
association
Relationships between Objects
1..∞
target_ID
identifier for an object type (s) of the Producer-Archive Project to which a relationship from this Transfer Object Type is established
relation_description
description of the relationship from this Descriptor's object type(s) to the target object type(s)
relation_descriptionType
relation_type
type of relation – for example "DED, Syntax, Context, Provenance, Reference, Fixity -
1..∞
relation_textual_description
free-text description of the relation
any
Optional (set of) attribute(s)
content
0..∞
Description of the actual Digital Objects described by this Descriptor.
contentType
data_object_ID
Logical Identifier for the described unit
data_object_occurrence
min_occurrence
integer value or 'unknown'
max_occurrence
integer value or 'unknown'
data_object_description
Textual description of the Data Object and its principal characteristics
data_object_format
Data Object type: pdf ...
mime_type
attributes
other_format
format_name
format_description
data_object_association
1..∞
target_ID
identifier for an object type of the Producer-Archive Project to which a relationship from this Data Object type is established
relation_description
relation_descriptionType
relation_type
type of relation – for example "DED, Syntax, Context, Provenance, Reference, Fixity -
1..∞
relation_textual_description
free-text description of the relation
any
Optional (set of) attribute(s)
content
0..∞
any
Optional (set of) attribute(s)
Generated with XMLSpy Schema Editor www.xmlspy.com

Figure 15: Generic Transfer Object Descriptor Model XML Schema

4.2.1.2 Generic Collection Descriptor Model

Figure 16 shows the generic Collection Descriptor Model. It is implemented with an XML Schema (see Annex A.2 for the XML source code).

This schema contains the attributes specified in section 3.4.4.2.2 "Collection Descriptor".
It is based on a hierarchical structure identifying the main descriptive parts of the collection. The schema itself contains comments on each field. It is divided in 4 parts (that can be divided in their turn) « identification, description, relation, any»:

- **identification**: cf. « Identification » section 3.4.4.2.2
- **description**: cf. "Collection description" section 3.4.4.2.2.
- **relation**: cf. "Relationships within the POT" section 3.4.4.2.2.
- **any**: cf . « User Define Attributes » section 3.4.4.2.2. This element may contain itself a sequence of elements (or another sequence) depending on the Producer-Archive Project needs.

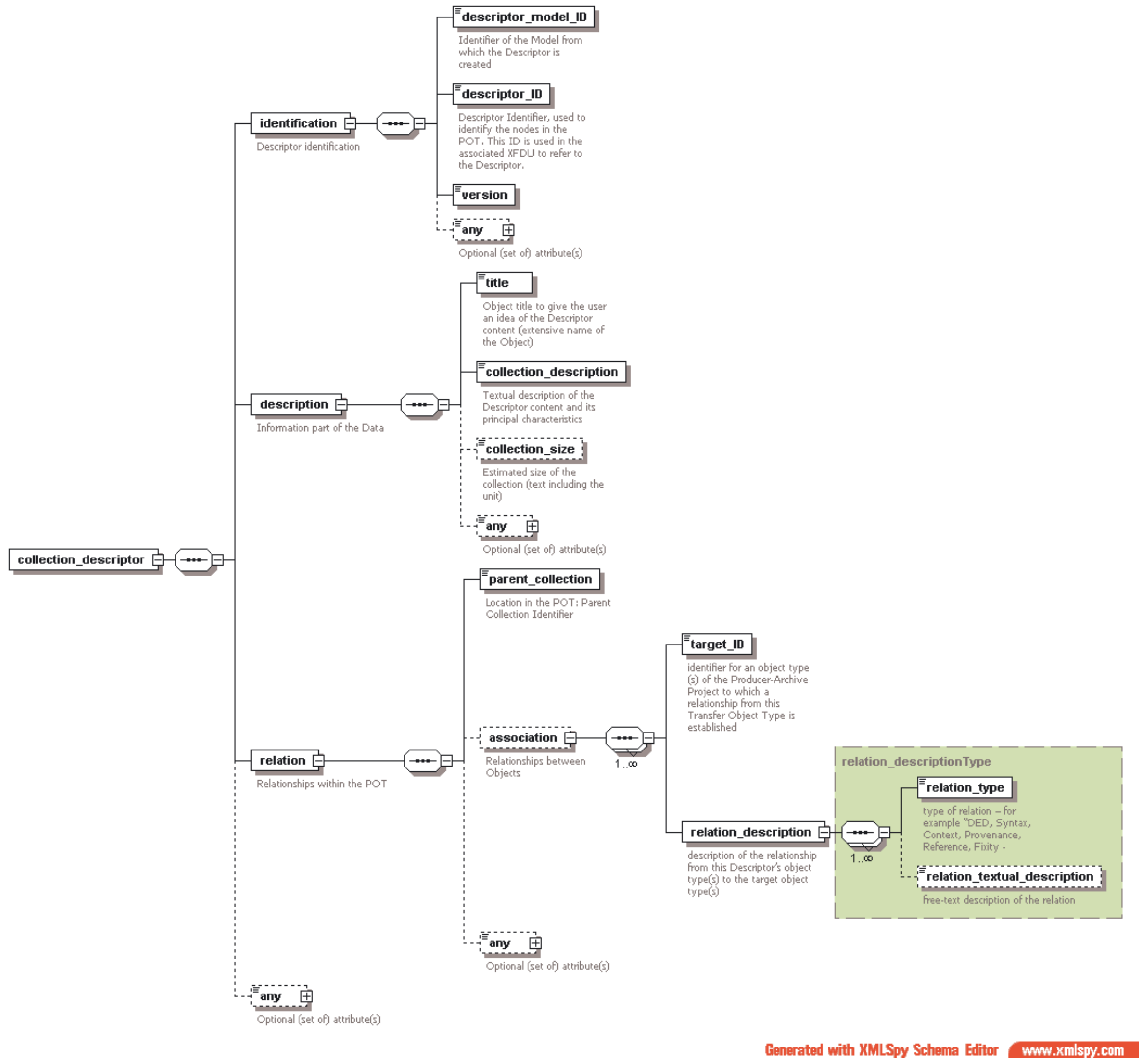

descriptor_model_ID
Identifier of the Model from which the Descriptor is created
descriptor_ID
Descriptor Identifier, used to identify the nodes in the POT. This ID is used in the associated XFDU to refer to the Descriptor.
identification
Descriptor identification
version
any
Optional (set of) attribute(s)
title
Object title to give the user an idea of the Descriptor content (extensive name of the Object)
collection_description
Textual description of the Descriptor content and its principal characteristics
description
Information part of the Data
collection_size
Estimated size of the collection (text including the unit)
any
Optional (set of) attribute(s)
collection_descriptor
parent_collection
Location in the POT: Parent Collection Identifier
target_ID
identifier for an object type (s) of the Producer-Archive Project to which a relationship from this Transfer Object Type is established
association
Relationships between Objects
1..∞
relation
Relationships within the POT
relation_descriptionType
relation_type
type of relation – for example "DED, Syntax, Context, Provenance, Reference, Fixity -
relation_description
description of the relationship from this Descriptor's object type(s) to the target object type(s)
1..∞
relation_textual_description
free-text description of the relation
any
Optional (set of) attribute(s)
any
Optional (set of) attribute(s)
Generated with XMLSpy Schema Editor www.xmlspy.com

Figure 16: Generic Collection Descriptor Model XML Schema

4.2.2 GENERIC DESCRIPTOR SPECIALIZATION

The purpose here is to create the Descriptor Models adapted to the Producer-Archive Project.
These Descriptor Models are themselves XML schema derived from the generic XML schema previously defined.
The Producer and the Archive should both access these XML schemas (copied, or one repository deposed in a shared space).

The generic Descriptor can be specialized by:

- suppression of optional elements ;
- modification of occurrence numbers (for example 10..15 instead of 1..n);
- addition of new elements (or sequence of elements) by the use of the groups « any »;
- definition of a list (enumerate) for the elements " descriptor_model_ID" or "relation_type";
- definition of patterns for identifiers (imposed nomenclature) or restriction of string length.

The creation of the Descriptor Models can be simplified by using an XML editor (e.g. XML Spy).
This kind of tool enables XML schemas manipulation in a graphic (and convivial) way without need for strong XML knowledge.

At the end, the set of Descriptor Models forms the Project Data Dictionary. They are supposed to be gathered in the same repository. They must be accessible for the people in charge of the creation of Descriptors.

4.2.3 DESCRIPTORS

This is the final stage for building the POT and the representation of this Plan according to needs.
The Descriptors, used to build the POT, are XML files. They are based on the specialized schemas defined in the previous stage.

In principle it is the Producer task to instantiate the Descriptors because it is supposed to know best the object information. These Descriptors can be instantiated in several times during the Formal Definition Phase (all information is not necessarily available at the same time). Moreover, different and distant people may access and instantiate the Descriptors (via a distributed application system). Nevertheless there should be One person responsible for the POT (receives, gathers and manages all the XML instances).

The persons filling the Descriptors may not be expert in XML. This is why it is possible to use forms proposed by XML editors (e.g. XML Spy or Xample) to create the Descriptors and to check that the produced XML file is conform to the initial XML schema. In case the XML file is produced in several times and with an XML editor, usually all the mandatory fields should have been filled to check the conformity[14].

Each Descriptor is linked:

- to the type of objects via the « descriptor_model_ID » (name given to the XML schema);
- to the XML schema (access path and file name) in the header of the XML file;
- to the other objects of the POT via the node identifiers (« relation » element).

Figure 17 is an example of a form used to describe a node using Xample editor (using values of figure 8). A short description of the attributes, which is a more 'user-friendly', replaces the attribute names. One might also imagine a set of formulas written in HTML describing each of the nodes in the POT.

[14]*If no tool is available to instantiate the Descriptors, it is always possible to generate a conform template from a Descriptor, and to fill it with a text editor.*

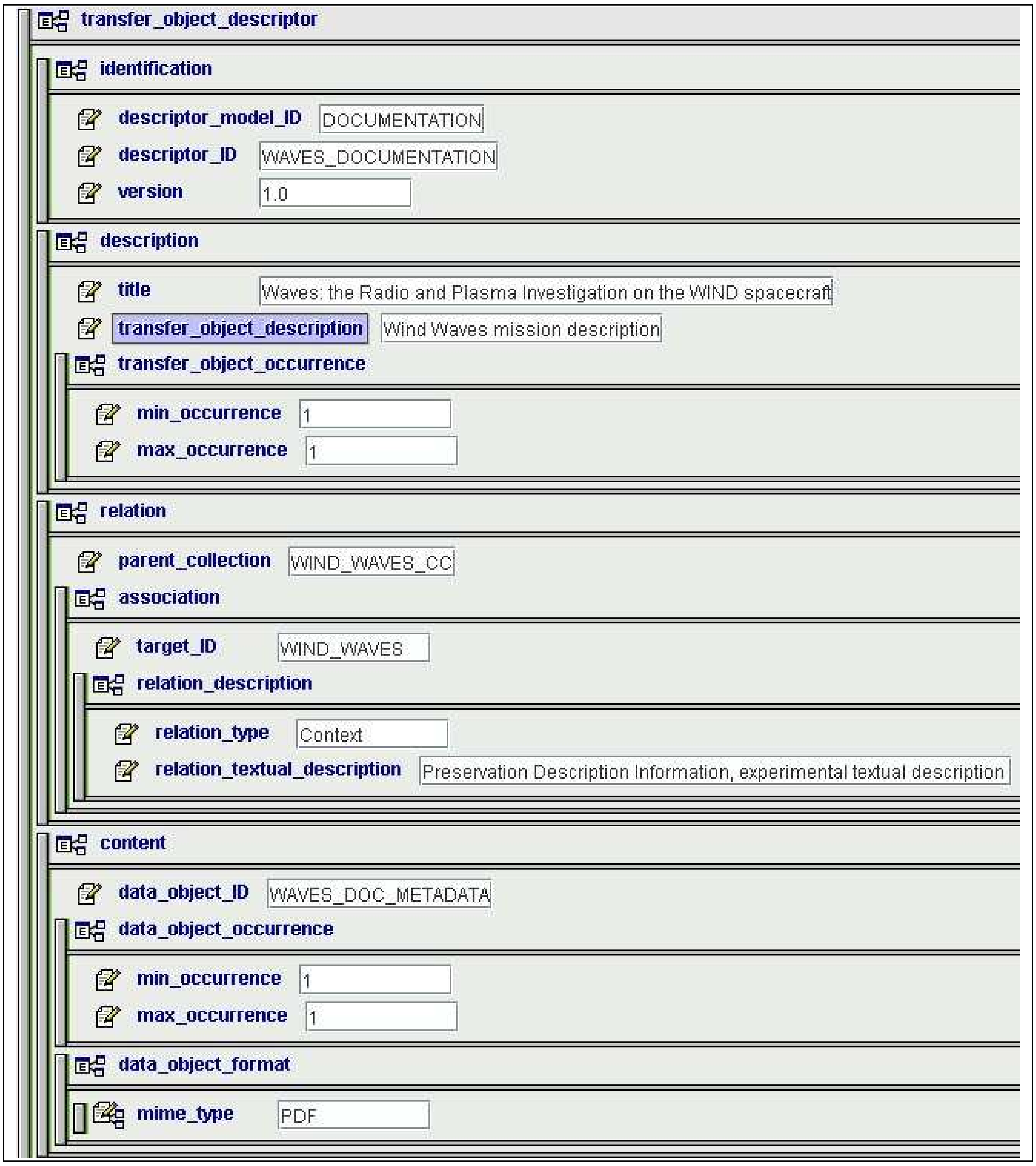

transfer_object_descriptor
identification
descriptor_model_ID DOCUMENTATION
descriptor_ID WAVES_DOCUMENTATION
version 1.0
description
title Waves: the Radio and Plasma Investigation on the WIND spacecraft
transfer_object_description Wind Waves mission description
transfer_object_occurrence
min_occurrence 1
max_occurrence 1
relation
parent_collection WIND_WAVES_CC
association
target_ID WIND_WAVES
relation_description
relation_type Context
relation_textual_description Preservation Description Information, experimental textual description
content
data_object_ID WAVES_DOC_METADATA
data_object_occurrence
min_occurrence 1
max_occurrence 1
data_object_format
mime_type PDF

Figure 17: Description of DOCUMENTATION node

4.2.4 POT IMPLEMENTATION AND VALIDATION

The POT implementation consists in building a base of the information contained in the Descriptors. This information can thus be accessed by the different functions of the Archive to insure the ingest function.

There are several POT validation levels to perform:

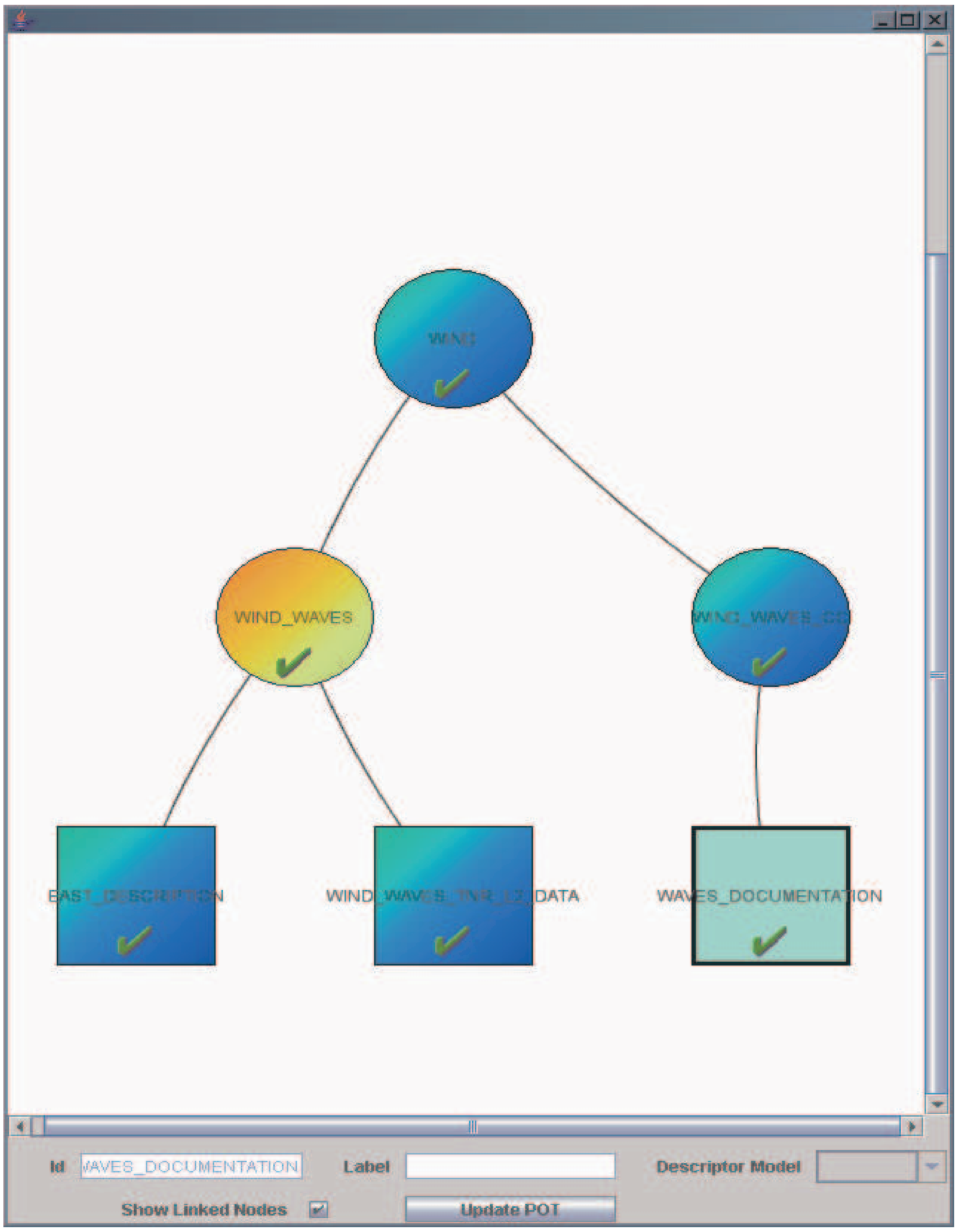
1. Validation of each XML instance with the associated XML schema (this can usually be done automatically by an XML capture tool or an XML editor).
2. Check the existence of the « descriptor_model_ID » (list of descriptor_model_IDs).
3. Global POT validation: Descriptors are inter dependent via the node identifiers. Check the coherence of the nodes:
    - No isolated node.
    - No duplicated descriptor_ID.
    - The nodes cited in the « relation » part must exist.
    - No ring.

4.2.5 POT VISUALISATION

The POT gives the Producer and the Archive a clear and non ambiguous vision of the expected objects.
The graphical representation of the POT shows at least:

- The type and number of expected objects (1..N if the number is unknown during the Formal Definition Phase).
- The links between these objects

Figure 18 is an example of the POT visualisation of the tree figure 6: the circle nodes show the Collections Descriptors, the square nodes show the Transfer Object Descriptors. The node "WAVES_DOCUMENTATION" is selected (light blue), and its associated node "WIND_WAVES" appears in yellow.



Id /AVES_DOCUMENTATION

Label

Descriptor Model

Figure 18: example of POT visualisation

4.3 XFDU IMPLEMENTATION OF SIP

The generic SIP Model implementation is based on XFDU (see [5]).

Figure 22 shows the process detailed in the following:

- Specialization of the XFDU standard towards a schema adapted to the SIP needs. The XML schema thus created applies to all types of SIPs.
- The future SIPs to be delivered during the Transfer Phase will be created from this Model. A SIP is described by an XFDU package complying with the XFDU standard.

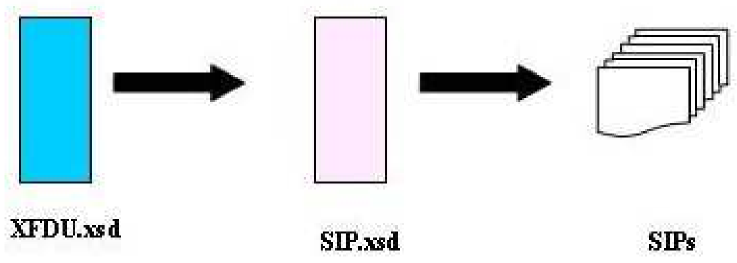


Figure 22: generic SIP Model process

Document [5] gives the generic XFDU XML schema.
An XFDU is a zip file including a “manifest” (XML file) and a set of Digital Objects. The manifest contains all the information describing the package content and how to access to the delivered objects (or referenced objects). The XFDU organization will not be described in further detail here.

The way to specialize this generic XFDU XML schema to take into account all the SIP needs will be completed later:

- specialization of the XFDU XML schema to tale into account the SIP information SIP (SIP global information, links between the Digital Objects and the Descriptors of the objects actually expected in the POT);
- how to take into account big files that must be split over several XFDUs (but belonging to the same SIP).

An extract of the XML schema will also be provided in annex A.2.

List of the specific SIP elements to take into account in an XFDU:

- Global information:
  - sip_ID;
  - producer_ID;
  - project_ID;
  - sip_type_ID;
  - sip_type_location;
  - sip sequence number;

  - o XFDU_sequence_position_in_sip;
  - o XFDU_sequence_size_in_sip.
- Information associated with each embedded object:
  - o descriptor_ID;
  - o descriptor_location;
  - o data_object_ID(s);
  - o last_object;
  - o updated_object.

4.4 SIP GROUPING CONSTRAINTS IMPLEMENTATION

The grouping constraints described in section 3.6.3 “SIP grouping constraints definition”, are defined in an XML file that conforms to the XML schema below. We recommend to use the following simple XML schema to implement the list of authorized Descriptors inside SIPs (see figure below).
Nota: the min_occurrence and max_occurrence attributes are implemented the same way as for the same attributes in the Transfer Object Descriptor.

This file will be used later during the Transfer Phase to build the SIPs.

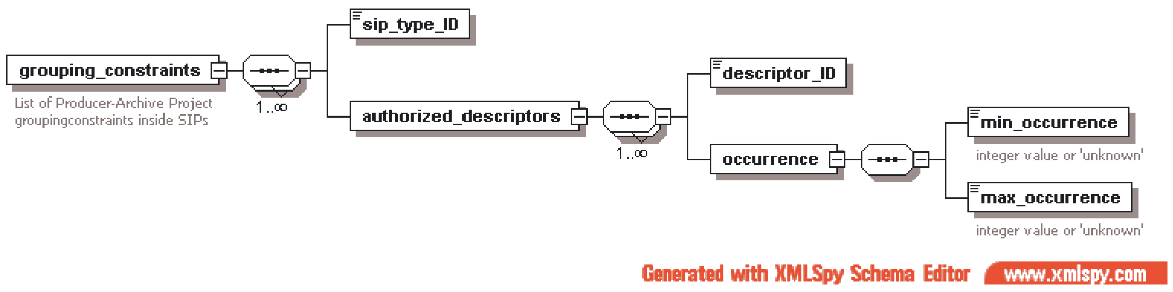

sip_type_ID
grouping_constraints
List of Producer-Archive Project groupingconstraints inside SIPs
1..∞
authorized_descriptors
descriptor_ID
1..∞
occurrence
min_occurrence
integer value or 'unknown'
max_occurrence
integer value or 'unknown'
Generated with XMLSpy Schema Editor
www.xmlspy.com

Figure 19: Grouping constraints XML schema

Example:
Figure 20 is an example of a grouping constraints table including 2 types of SIPs, “SIP1” and “SIP2” inspired from figure 6:
- “SIP1” contains 2 different Objects, an EAST description, and a Waves documentation (this type of SIP will be transferred once because there is only one object of each Descriptor to transfer in the project).
- “SIP2” contains one category of Object identified by “WIND_WAVES_TNR_L2_DATA”. A SIP of type SIP2 can contain several occurrences of this Object (this type of SIP will be transferred several times because there is many objects of this Descriptor to transfer in the project).

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by CNES (CNES) -->
<!--Sample XML file generated by XML Spy v4.3 U (http://www.xmlspy.com)-->
<grouping_constraints xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="L:\OAIS\SIPs\ConceptPaper\descripteurs\grouping_constraints.xsd">
```

```
    <sip_type_ID>SIP1</sip_type_ID>
    <authorized_descriptors>
        <descriptor_ID>EAST_DESCRIPTION</descriptor_ID>
        <occurrence>
            <min_occurrence>1</ min_occurrence >
            <max_occurrence>1</ max_occurrence >
       </occurrence>
        <descriptor_ID>WAVES_DOCUMENTATION</descriptor_ID>
        <occurrence>
            <min_occurrence>1</ min_occurrence >
            <max_occurrence>1</ max_occurrence >
      </occurrence>
    </authorized_descriptors>
    <sip_type_ID>SIP2</sip_type_ID>
    <authorized_descriptors>
        <descriptor_ID>WIND_WAVES_TNR_L2_DATA</descriptor_ID>
        <occurrence>1..N</occurrence>
    </authorized_descriptors>
</grouping_constraints>
```

Figure 20: example of grouping constraints table

4.5 SIP SEQUENCING CONSTRAINTS IMPLEMENTATION

The « time_constraint_group » and « serial_number_in_constraint » elements, described in section 3.6.4 "SIP Sequencing constraints definition", are defined in an XML file that conforms to the XML schema below. We recommend to use the following simple XML schema to implement the list of constraints between SIPs.

This file will be used later during the Validation Phase for each received SIP.

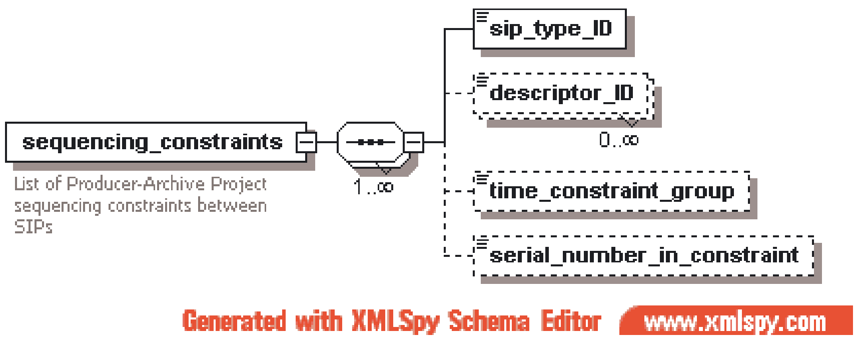

sequencing_constraints
List of Producer-Archive Project sequencing constraints between SIPs
1..∞
sip_type_ID
descriptor_ID
0..∞
time_constraint_group
serial_number_in_constraint
Generated with XMLSpy Schema Editor
www.xmlspy.com

Figure 21: Sequencing constraints XML schema

5. TRANSFER PHASE AND VALIDATION PHASE SPECIFICATION

Reminder: only the initial validation defined in the PAIMAS [1] is taken into account here. Hence, Transfer and Validation Phases are strongly nested.

5.1 SIP CREATION

It is necessary to automate the creation of the package if the production becomes important.

For each SIP to be transferred, the Producer has to:

- Give the SIP an identifier « SIP_ID » and a sequence number (if used) .
- Follow the grouping rules defined in the SIP grouping constraints table.
- Create the XML Manifest file with the needed SIP attributes, in particular with the link between the logical identifiers defined in the Descriptor(s) - « descriptor_ID, data_object_ID(s) » - and the transferred file names.
- From the generic SIP Model, create the zip file containing the XML Manifest and the Digital files to be transferred.

If the transferred object is the last of a collection of objects, the value of the "last_object" element is 1.
If the transferred file is a big one split over several SIPs, the elements needed for the reconstruction of the object must be instantiated.

Figure 23 sums up the input and output of the SIP creation.

Input		**Output**
Digital Objects	**SIP creation**	XFDUs including: The Digital Objects The following attributes: • sip_ID; • producer_ID; • project_ID; • sip_type_ID; • sip_type_location; • sip_sequence_number; • XFDU_sequence_position_in_sip; • XFDU_sequence_size_in_sip; • descriptor_ID; • descriptor_location; • data_object_ID(s); • last_object; • updated_object.
Generic SIP Model		
SIP grouping constraints		

Figure 23: SIP creation input and output

As soon as the SIP is ready, it can be transferred by the Producer via the defined communication procedure.

5.2 SIP TRANSFER: TRANSMISSION SESSIONS

Input	SIP transfer	Output
XFDUs by the Producer		XFDUs by the Archive
SIP sequencing constraints table		

Figure 24: SIP transfer input and output

Producer and Archive have agreed on a delivery schedule for the transfer during the Formal Definition Phase, depending on the planned production of the data.

Inside this schedule, the flow of the transferred SIPs is strongly linked to the Producer-Archive Project context: punctual, periodic, or continuous.

The delivery is organised in transmission sessions. A transmission session contains one or several SIPs. There is no constraint defined in this document on the content of a transmission session (types and number of SIPs), nor on the period of time of a transmission session.
It's up to the Archive and the Producer to agree on the best SIP transfer procedure and transmission session definition. This is part of the Submission Agreement.

For example:

- *In the case of space data produced every day, the Producer may decide to transfer only once a month the production in a single transmission session.*
- *Transmission sessions can be automated and periodically delivered in a deposit place defined by the Producer and the Archive.*
- *A continuous delivery of objects can be seen as one single transmission session, beginning with the 1st object delivery, and ending with the last one delivery, and this can cover several months.*

A transmission session is delimited by a beginning message "Transmission session start", and an ending message "Transmission session end", from the Producer towards the Archive (see section 5.5 'Message exchange definition').

It is possible for the Producer to send the Archive a "Transfer authorization request" message to inform that he's ready for the transfer. In this case the Archive replies with a "Transfer authorization reply" message.

If the Producer does not inform the Archive of each transfer, that means that Producer and Archive have agreed on a specific procedure. In some cases of automated delivery, the Archive regularly scrutinizes a deposit repository to detect data arrival.

5.3 TRANSFER FOLLOW-UP BY THE PRODUCER AND THE ARCHIVE

The Archive should have at any moment a clear vision of the progression of the Transfer Phase. The Producer should have a view on this follow-up. The way this follow-up is performed may vary from one Project to another. This paragraph gives a non-mandatory and possible view of this function.

The follow-up may include the following:

- log update,
- update of the number of delivered and validated Transfer Objects (compared to the expected ones),
- exchange and management of formal messages (see the following paragraph),
- progression of the transfer according to the schedule,
- etc.

The log contains at least information on transmission sessions (session reference, start and end date, content of the session), and on anomalies detected (anomaly identifier, erroneous function, erroneous file, …).

The Archive updates the number of delivered and validated Transfer Objects by the use of two indicators:

- **Status**: status of the object during the transfer: expected (no object delivered), pending (case of a collection whose object delivery spreads over a time period), closed (no more objects expected).
- **Number_validated_objects**: number of objects already received and validated. This number is compared to the "transfer_object_occurrence".

  Reminder: "transfer_object_occurrence" (number of objects with the same« descriptor_ID » initially expected) is a static information known during the Formal Definition Phase (contained in the Transfer Object Descriptor).

These indicators are updated after each validated SIP. With the Transfer Object Type (« descriptor_ID ») and the node in the POT (« parent_collection »), they may be used to graphically follow the progress of the Transfer Phase.

5.4 SIP RECEPTION AND VALIDATION

On delivery (see 5.2), the Archive has to check first arrival of transferred SIPs, and then the SIPs content.
The Archive may move the SIPs in a validation repository.

The Archive:

- Checks the arrival of SIPs via the SIP sequence number. This is the recommended way to test that the SIPs have all arrived.
- The Archive checks in particular that the inter-dependant SIPs have arrived (one SIP split over several packages) via the sub sequence number.
- Sends the Producer -"Producer_ID"- an ackowledgement of receipt –"Transfer acknowledgement" message- (this acknowledgement can be sent later with the validation message once the validation has been performed).

The Archive then recovers the information contained in the SIPs, and performs initial validations (respect for the grouping and sequencing constraints, compliance with the POT). Thus, the Archive follows the process:

1. Dezip each XFDU, and for each XFDU:

2. Opens the Manifest ;
3. Gets the « sip_type_ID » value, the file location, and checks:
    a. If the SIP contains the right objects (comparing to the table of grouping constraints).
    b. If the SIP respects the sequencing constraints by comparison between the SIP « sip_type_ID » and the « sip_type_ID » of already transferred and validated SIPs.
4. Checks that each embedded object is an expected one:
    a. Reads the value of each "descriptor_ID".
    b. Gets the value of "status" element, the number of objects already transferred and validated "number_validated_objects", and the number of expected objects "transfer_object_occurrence".
    c. Compares the values:

    for an expected object,

    "number_validated_objects" + 1 must be lower or equal to "transfer_object_occurrence".

    If the element "last_object'' = 1, then

    "number_validated_objects" + 1 must be equal to "transfer_object_occurrence"

    (except if this value was unknown at the beginning).
5. Reads the different embedded objects identifiers, compares these IDs with the corresponding data_object_ID(s) in the POT, and checks that all the mandatory information are present.

If an anomaly is detected (for example the file *file_name* does not exist), the Archive sends the Producer – "Producer_ID"- a message (see 5.5).

If no error has been detected during the validation, the Archive sends the Producer a "SIP validation notification" message (see section 5.5 « Message exchange definition »).

If an error occurs during the previous validation process for one object in a SIP:

- This SIP is rejected (the error applies to the whole SIP, even if this SIP contains other Transfer Objects that are valid).
- An anomaly message is sent to the Producer (see section 5.6 "Anomaly management"), with a text such as « the metadata file *metadata_filename* in SIP SIP_ID does not exist ».
- The follow-up indicators are not updated for all the objects of this SIP.

Figure 25 sums up the input and output of the SIP reception and validation.

Input		**Output**
XFDUs	**SIP reception and validation**	Digital Objects validated
The following attributes (from Transfer Object Descriptors and SIP grouping and sequencing constraints tables): • sip_type_ID and location • occurrence • time_constraint_group • serial_number_in_constraint • descriptor_ID (1..n) and location		

• data_object_ID (1..n)		

Figure 25: SIP reception and validation input and output

See section « Change management » if the delivered object was an already archived one (“updated”). This means that this object may belong to a collection whose status is “closed”. This object will be checked on each point except point 4. If this object is validated, the indicators “status” and “number_validated_objects” won’t be updated (object already counted). There is no notion of object version in this document.

The process is the same for automated and periodic deliveries.

5.5 MESSAGE EXCHANGE DEFINITION

Archive and Producer have agreed during the Formal Definition Phase on a message exchange process, from the Archive to the Producer, or from the Producer to the Archive: how and when to communicate on events, definition of structure and content of messages.
This process could be more or less complex and partly automated. The list of messages given here is not exhaustive. Except the validation and anomaly messages, the messages are optional.
This standard doesn’t recommend an implementation formalism for this message exchange.

Messages from the Producer to the Archive:

- **Transfer authorization request**: useful in the case of punctual delivery.
- **Transmission session start**: notification for the beginning of a transmission session.
- **Transmission session end**: notification for the end of a transmission session.

Messages from the Archive to the Producer:

- **Transfer authorization reply**: reply to the message « Transfer authorization request ».
- **Transfer acknowledgement**: this acknowledgement may be grouped with the following message « SIP validation notification ». Otherwise, it can be sent SIP per SIP, or for a group of SIPs, or for a whole transmission session.
- **SIP validation notification**: notification of acceptance of transferred SIP.
- **Anomaly notification**: anomaly message. An anomaly may occur at different levels of the transfer process, depending on the level of checks performed during this transfer.

Figure 26 shows the process of message exchange between the Producer and the Archive (in the form of an UML sequence diagram).

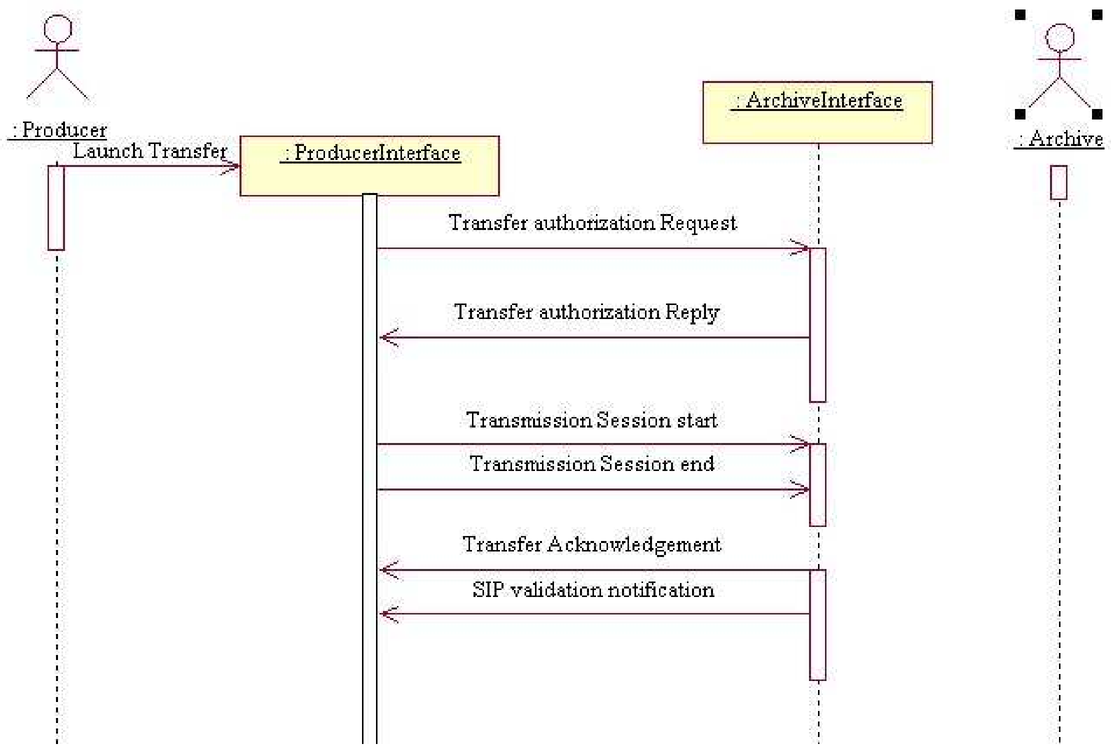


Figure 26: message exchange

A message is made up of a common header, and a body. The content of the body depends on the type of the message.

Header:

- Message_ID (M, 1..1): message identifier.
- Message_code (M, 1 ..1): type of the delivered message: Transfer authorization request, Archive transfer acknowledgement, …
- Producer_ID (M, 1..1): Producer identifier.
- Project_ID (M, 1..1): Producer-Archive Project identifier.
- Session_ID (O, 0..1): session identifier. Mandatory in "Transmission session start/end" messages.
- Date (M, 1..1): message date.

Body (in the following table, the first column is the message type, the second one is a description of the message body):

Type of message	Body of the message.
Transfer authorization request	Producer request. This message contains if necessary an evaluation of the volume of the data to transfer, and/or a deposit repository.
Transfer authorization reply	Archive reply to the previous message, including: • ID of the "Transfer authorization request" message (O, 0..1).

	• Name and address of a deposit repository (if necessary).
Transmission session start	The Producer gives the Archive the following information: • transfer_repository (O, 0..1): the location where the Archive can get the transferred files (for example a deposit repository on the Archive side). In the case of media sending, the "transfer_repository" is the reference of the sent media.
Transmission session end	The Producer gives the Archive the following information: • sip_list (M, 1..1): list of the transferred files of this session (file names, number of files). • checksum, file size or other useful indicator[15] (O, 0..1).
Transfer acknowledgement	The Archive gives the Producer the following information: • ID of the "Transmission session end " message (O, 0..1). • Notification of the list of received SIPs (SIP_IDs) (M, 1..1).
SIP validation notification	Done SIP per SIP. To simplify, the message may group a list of validated SIPs. The Archive gives the Producer the following information: • Validated SIP IDs (M, 1..N).
Anomaly notification	For each detected anomaly, the Archive sends an anomaly message to the Producer with the following information: • anomaly_ID (M, 1..1): anomaly identifier. • error_level (O, 0..1): error level (from serious to informative). • validation_function (M, 1..1): function where the anomaly has been detected (SIPs validation, …). • error_number (O, 0..1). • text (M, 1..1): error explanation (with reference to the erroneous object).

5.6 ANOMALY MANAGEMENT

During the Formal Definition Phase, the Producer and the Archive have defined the initial validation performed on the received SIPs.
For each detected anomaly, the Archive:

- Sends an anomaly message to the Producer (see section 5.5 for the message content).
- Performs the actions linked to the type of anomaly (for example object rejection in the case of an erroneous transferred object).

The Archive holds the anomaly list used by the anomaly function. The consequences of an anomaly depends on the level of error and the validation function concerned.

[15]*If not already taken into account by the XFDU itself.*

6. MANAGING MODIFICATIONS

This includes all modifications which occur once the Submission Agreement has been approved. Some modifications may lead to a re-negotiation of the Submission Agreement (see document [1], section 3.2.2.6).

There are two categories of modifications:

- Category 1: The POT modifications: modification of Model dictionaries and Descriptors. These modifications should be done in compliance with the rules defined in the PAIMAS. The rules for managing these modifications may moreover have been specified in the Submission Agreement.

These modifications concern for instance:

- The addition or suppression of a collection of objects.
- The addition or suppression of types of Transfer Objects.
- The modification of Collection or Transfer Object Descriptors which have already been defined.
- The addition, suppression or modification of elements of dictionary elements and the consequences on the POT negotiated for the Archive Project.

- Category 2: data modification, or data error detected after the initial validation.

6.1 POT MODIFICATION

The dictionary may evolve during the Producer-Archive Project.

Each modification on a Descriptor Model may impact the existing Descriptors and the future ones.
Each modification on a Descriptor may impact the POT and the tools using the POT.
In each case, the impact on the existing Descriptors and the POT on the one hand, on the POT and dictionary tools on the other hand, should be analyzed (list not exhaustive):

- Impact on the data to be delivered.
- Technical impact on the current functions and tools.
- Suppression of objects already delivered.
- Possible re-negotiation of the Submission Agreement.

Here are the possible POT modifications:

1 Dictionary modification by the addition of a new Descriptor Model, and addition of new Descriptors associated with this new Model: a possible impact is the creation of a new SIP Type (or the modification of an existing one). This implies:
   - Dictionary update.
   - List of Descriptor types and SIP Types update.
   - POT update and visualisation update (new descriptor_ID and new type of information), POT validation.
   - The new attributes are inserted in the different system functions (Formal Definition Phase, Transfer and Validation Phases).

2 Dictionary modification by the modification of an existing Descriptor Model: this may impact the existing corresponding Descriptor, and thus may imply a new version of these Descriptors:
   - Update of existing Descriptors (and existing links) by using the Descriptor Model “version”

element. The descriptor_ID and the links towards it remain unchanged[16].

3 No dictionary modification, addition of a new Descriptor of an existing Descriptor Model: same as case 1.

4 No dictionary modification, modification of an existing Descriptor: same as case 2.

6.2 OBJECT MODIFICATION

The Producer indicates the transfer of an object already transferred and validated by the element « **updated** » in the Manifest of the SIP. The reception and validation process remains the same. The processing of the follow-up indicators will be different because the object has already be counted in a previous delivery.
We assume that the new object replaces the older one (all the links towards this object remain the same).
It's up to the Archive to decide to keep a backup of the versions of the objects already delivered.

[16] *Another solution is to change the descriptor_ID value. It is then considered as a new object (case 1).*

7. TOOLS

The Producer and the Archive should identify the needs for tools at the outset and should evaluate development efforts if the tools have to be developed or adapted.

The implementation suggested in this document, is based on XML Schemas. The practical use of this implementation by an Archive Project involves setting up a certain number of tools for this implementation. The list given below is not exhaustive.

The Formal Definition Phase:

- Tool for creating specialized dictionaries from the generic Models proposed (a tool such as XML Spy can be used to specialize an XML Schema).
- Tool for generating Descriptors. For the information acquisition, it is possible to use forms generating XML files from the initial XML Schemas. This acquisition should be done by different persons and in several times.
- Tool for creating and validating the POT. The tool should :
  1. Create the POT from the dictionary (Descriptor Models). To do this, the information contained in the Descriptors are inserted in an ingest base. This base will provide a complete view on the nodes of the POT and the links between these nodes.
  2. Validate the POT by checking the coherence between all the described elements (identifiers, links, …).
  3. The previous tool should be completed to produce a graphic and written representation of the POT which can be understood by the Producer and the Archive. This representation will be useful to facilitate the dialog between the Archive and the Producer. This representation will also be used for the follow-up of the Transfer Phase.

The Transfer Phase:

- Tool for generating SIP instances (for the moment we are waiting for a prototype to create XFDU instances):
  - Software generation when there are many instances.
  - Generation by forms for the other cases.

- Tool for the transfer validation and management. The tool should:
  1. Validate the content of each received SIP according to the initial validation plan (object expected and conforms to the associated Descriptor).
  2. Check that the order of SIP arrival respects the sequencing constraints.
  3. Monitor project progress for the Producer and for the Archive, taking into account: the management of the log of deliveries, the update of the follow-up indicators.
  4. Propose a follow-up graphical view by using, if possible, the POT graphical view. This view should be shared by the Producer and the Archive.

ANNEX A

(This annex **is** part of the Recommendation)

A.1 GENERIC TRANSFER OBJECT DESCRIPTOR MODEL

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 U (http://www.xmlspy.com) by cnes (CENTRA NATIONAL D ETUDES SPATIALES) -->
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by CNES (CNES) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">

	<xs:element name="transfer_object_descriptor">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="identification">
					<xs:annotation>
						<xs:documentation>Descriptor identification</xs:documentation>
					</xs:annotation>
					<xs:complexType>
						<xs:sequence>
							<xs:element name="descriptor_model_ID" type="xs:string">
								<xs:annotation>
									<xs:documentation>Identifier of the Model from which the Descriptor is
created</xs:documentation>
								</xs:annotation>
							</xs:element>
							<xs:element name="descriptor_ID" type="xs:string">
								<xs:annotation>
									<xs:documentation>Descriptor Identifier, used to identify the nodes in the POT. This ID is used in
the associated XFDU to refer to the Descriptor.</xs:documentation>
								</xs:annotation>
							</xs:element>
							<xs:element name="version" type="xs:string"/>
							<xs:element name="any" type="xs:anyType" minOccurs="0">
								<xs:annotation>
									<xs:documentation>Optional (set of) attribute(s)</xs:documentation>
								</xs:annotation>
							</xs:element>
						</xs:sequence>
					</xs:complexType>
				</xs:element>
				<xs:element name="description">
					<xs:annotation>
						<xs:documentation>Information part of the Data</xs:documentation>
					</xs:annotation>
					<xs:complexType>
						<xs:sequence>
							<xs:element name="title" type="xs:string">
								<xs:annotation>
									<xs:documentation>Object title to give the user an idea of the Descriptor content (extensive name
of the Object)</xs:documentation>
								</xs:annotation>
							</xs:element>
							<xs:element name="transfer_object_description" type="xs:string">
								<xs:annotation>
									<xs:documentation>Textual description of the Descriptor content and its principal
characteristics</xs:documentation>
								</xs:annotation>
							</xs:element>
							<xs:element name="transfer_object_occurrence">
								<xs:complexType>
									<xs:sequence>
										<xs:element name="min_occurrence" type="SIMPLE_TYPE_UNION">
```

```
                                            <xs:annotation>
                                                <xs:documentation>integer value or 'unknown'</xs:documentation>
                                            </xs:annotation>
                                        </xs:element>
                                        <xs:element name="max_occurrence" type="SIMPLE_TYPE_UNION">
                                            <xs:annotation>
                                                <xs:documentation>integer value or 'unknown'</xs:documentation>
                                            </xs:annotation>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="transfer_object_size" type="xs:string" minOccurs="0">
                                <xs:annotation>
                                    <xs:documentation>Estimated size of the Data Objects (text including the
unit)</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                            <xs:element name="any" type="xs:anyType" minOccurs="0">
                                <xs:annotation>
                                    <xs:documentation>Optional (set of) attribute(s)</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="relation">
                    <xs:annotation>
                        <xs:documentation>Relationships within the POT</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="parent_collection" type="xs:string">
                                <xs:annotation>
                                    <xs:documentation>Location in the POT: Parent Collection Identifier </xs:documentation>
                                </xs:annotation>
                            </xs:element>
                            <xs:element name="association" minOccurs="0">
                                <xs:annotation>
                                    <xs:documentation>Relationships between Objects</xs:documentation>
                                </xs:annotation>
                                <xs:complexType>
                                    <xs:sequence maxOccurs="unbounded">
                                        <xs:element name="target_ID" type="xs:string">
                                            <xs:annotation>
                                                <xs:documentation>identifier for an object type (s) of the Producer-Archive Project to
which a relationship from this Transfer Object Type is established</xs:documentation>
                                            </xs:annotation>
                                        </xs:element>
                                        <xs:element name="relation_description" type="relation_descriptionType">
                                            <xs:annotation>
                                                <xs:documentation>description of the relationship from this Descriptor’s object
type(s) to the target object type(s)</xs:documentation>
                                            </xs:annotation>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="any" type="xs:anyType" minOccurs="0">
                                <xs:annotation>
                                    <xs:documentation>Optional (set of) attribute(s)</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="content" type="contentType" minOccurs="0" maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>Description of the actual Digital Objects described by this Descriptor. </xs:documentation>

                    </xs:annotation>
                </xs:element>
                <xs:element name="any" type="xs:anyType" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>Optional (set of) attribute(s)</xs:documentation>
                    </xs:annotation>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:simpleType name="SIMPLE_TYPE_UNION">
        <xs:union memberTypes="xs:nonNegativeInteger">
            <xs:simpleType>
                <xs:restriction base="xs:NMTOKEN">
                    <xs:enumeration value="unknown"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:union>
        <!-- entier ou unknown -->
    </xs:simpleType>
    <xs:complexType name="contentType">
        <xs:sequence>
            <xs:element name="data_object_ID" type="xs:string">
                <xs:annotation>
                    <xs:documentation>Logical Identifier for the described unit</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="data_object_occurrence">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="min_occurrence" type="SIMPLE_TYPE_UNION">
                            <xs:annotation>
                                <xs:documentation>integer value or 'unknown'</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                        <xs:element name="max_occurrence" type="SIMPLE_TYPE_UNION">
                            <xs:annotation>
                                <xs:documentation>integer value or 'unknown'</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="data_object_description" type="xs:string" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>Textual description of the Data Object and its principal characteristics</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="data_object_format">
                <xs:annotation>
                    <xs:documentation>Data Object type: pdf ...</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                    <xs:choice>
                        <xs:element name="mime_type">
                            <xs:complexType>
                                <xs:simpleContent>
                                    <xs:extension base="xs:string">
                                        <xs:attribute name="list_mime_type"/>
                                    </xs:extension>
                                </xs:simpleContent>
                            </xs:complexType>
                        </xs:element>
                        <xs:element name="other_format">
                            <xs:complexType>
                                <xs:sequence>
```

```
                                    <xs:element name="format_name" type="xs:string"/>
                                    <xs:element name="format_description" type="xs:string"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
            <xs:element name="data_object_association" minOccurs="0">
                <xs:complexType>
                    <xs:sequence maxOccurs="unbounded">
                        <xs:element name="target_ID">
                            <xs:annotation>
                                <xs:documentation>identifier for an object type of the Producer-Archive Project to which a relationship
from this Data Object type is established</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                        <xs:element name="relation_description" type="relation_descriptionType"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="any" type="xs:anyType" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>Optional (set of) attribute(s)</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="content" type="contentType" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="relation_descriptionType">
        <xs:sequence maxOccurs="unbounded">
            <xs:element name="relation_type" type="xs:string">
                <xs:annotation>
                    <xs:documentation>type of relation – for example “DED, Syntax, Context, Provenance, Reference, Fixity -
</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="relation_textual_description" type="xs:string" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>free-text description of the relation</xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

(This annex **is** part of the Recommendation)

A.2 GENERIC COLLECTION DESCRIPTOR MODEL

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 U (http://www.xmlspy.com) by cnes (CENTRA NATIONAL D ETUDES SPATIALES) -->
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by CNES (CNES) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">

    <xs:element name="collection_descriptor">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="identification">
                    <xs:annotation>
                        <xs:documentation>Descriptor identification</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="descriptor_model_ID" type="xs:string">
                                <xs:annotation>
                                        <xs:documentation>Identifier of the Model from which the Descriptor is
created</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                            <xs:element name="descriptor_ID" type="xs:string">
                                <xs:annotation>
                                        <xs:documentation>Descriptor Identifier, used to identify the nodes in the POT. This ID is
used in the associated XFDU to refer to the Descriptor.</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                            <xs:element name="version" type="xs:string"/>
                            <xs:element name="any" type="xs:anyType" minOccurs="0">
                                <xs:annotation>
                                        <xs:documentation>Optional (set of) attribute(s)</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="description">
                    <xs:annotation>
                        <xs:documentation>Information part of the Data</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="title" type="xs:string">
                                <xs:annotation>
                                        <xs:documentation>Object title to give the user an idea of the Descriptor content (extensive
name of the Object)</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                            <xs:element name="collection_description" type="xs:string">
                                <xs:annotation>
                                        <xs:documentation>Textual description of the Descriptor content and its principal
characteristics</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                            <xs:element name="collection_size" type="xs:string" minOccurs="0">
                                <xs:annotation>
                                        <xs:documentation>Estimated size of the collection (text including the
unit)</xs:documentation>
                                </xs:annotation>
                            </xs:element>
                            <xs:element name="any" type="xs:anyType" minOccurs="0">
                                <xs:annotation>
                    <xs:documentation>Optional (set of) attribute(s)</xs:documentation>
                </xs:annotation>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="relation">
          <xs:annotation>
            <xs:documentation>Relationships within the POT</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element name="parent_collection" type="xs:string">
                <xs:annotation>
                    <xs:documentation>Location in the POT: Parent Collection Identifier </xs:documentation>

                </xs:annotation>
              </xs:element>
              <xs:element name="association" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>Relationships between Objects</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                    <xs:sequence maxOccurs="unbounded">
                         <xs:element name="target_ID" type="xs:string">
                              <xs:annotation>
                                   <xs:documentation>identifier for an object type (s) of the
Producer-Archive Project to which a relationship from this Transfer Object Type is established</xs:documentation>
                              </xs:annotation>
                         </xs:element>
                         <xs:element name="relation_description" type="relation_descriptionType">
                              <xs:annotation>
                                   <xs:documentation>description of the relationship from this
Descriptor's object type(s) to the target object type(s)</xs:documentation>
                              </xs:annotation>
                         </xs:element>
                    </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="any" type="xs:anyType" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>Optional (set of) attribute(s)</xs:documentation>
                </xs:annotation>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="any" type="xs:anyType" minOccurs="0">
          <xs:annotation>
            <xs:documentation>Optional (set of) attribute(s)</xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:simpleType name="SIMPLE_TYPE_UNION">
    <xs:union memberTypes="xs:int xs:string"/>
  </xs:simpleType>
  <xs:complexType name="contentType">
    <xs:sequence>
      <xs:element name="data_object_ID" type="xs:string">
        <xs:annotation>
          <xs:documentation>Logical Identifier for the described unit</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="data_object_occurrence" type="SIMPLE_TYPE_UNION"/>
      <xs:element name="data_object_description" type="xs:string" minOccurs="0">
        <xs:annotation>
```

```
					<xs:documentation>Textual description of the Data Object and its principal characteristics</xs:documentation>
				</xs:annotation>
			</xs:element>
			<xs:element name="data_object_format">
				<xs:annotation>
					<xs:documentation>Data Object type: binary ...</xs:documentation>
				</xs:annotation>
				<xs:complexType>
					<xs:choice>
						<xs:element name="mime_type" type="xs:string"/>
						<xs:element name="other_format" type="xs:string"/>
					</xs:choice>
				</xs:complexType>
			</xs:element>
			<xs:element name="data_object_association" minOccurs="0">
				<xs:complexType>
					<xs:sequence maxOccurs="unbounded">
						<xs:element name="target_ID"/>
						<xs:element name="relation_description" type="relation_descriptionType"/>
					</xs:sequence>
				</xs:complexType>
			</xs:element>
			<xs:element name="content" type="contentType" minOccurs="0" maxOccurs="unbounded"/>
		</xs:sequence>
	</xs:complexType>
	<xs:complexType name="relation_descriptionType">
		<xs:sequence maxOccurs="unbounded">
			<xs:element name="relation_type" type="xs:string">
				<xs:annotation>
					<xs:documentation>type of relation – for example “DED, Syntax, Context, Provenance, Reference, Fixity -
</xs:documentation>
				</xs:annotation>
			</xs:element>
			<xs:element name="relation_textual_description" type="xs:string" minOccurs="0">
				<xs:annotation>
					<xs:documentation>free-text description of the relation</xs:documentation>
				</xs:annotation>
			</xs:element>
		</xs:sequence>
	</xs:complexType>
</xs:schema>
```

(This annex **is** part of the Recommendation)

A.3 GENERIC SIP MODEL

This part will be completed later according to the XFDU standard with the SIP needs.

ANNEX B : INFORMATIVE REFERENCES

(This annex **is not** part of the Recommendation)

This annex provides a list of references that may be valuable to the user of this Recommendation as background material or to provide implementation guidelines for using this Recommendation.

[B1] *Procedures Manual for the Consultative Committee for Space Data Systems.* CCSDS A00.0-Y-9. Yellow Book. Issue 9. Washington, D.C.: CCSDS, November 2003.

[B2] *Data Entity Dictionary Specification Language (DEDSL)—Abstract Syntax (CCSD0011).* Recommendation for Space Data System Standards, CCSDS 647.1-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, June 2001. [Equivalent to ISO 21961:2003.]

[B3] *Data Entity Dictionary Specification Language (DEDSL)— XML/DTD Syntax (CCSD0013).* Recommendation for Space Data System Standards, CCSDS 647.3-B-1. Blue Book. Issue 1. Washington, D.C.: CCSDS, January 2002. [Equivalent to ISO 22643:2003.]

[B4] *The Data Description Language EAST Specification— (CCSD0010).* Recommendation for Space Data System Standards, CCSDS 644.0-B-2. Blue Book. Issue 2. Washington, D.C.: CCSDS, November 2000. [Equivalent to ISO 15889:2003.]

[B5] *FGDC Standards Reference Model.* Washington, DC: Federal Geographic Data Committee, March 1996. http://www.fgdc.gov/standards/refmod97.pdf

[B6] *Dictionary of Archival Terminology: English and French with Equivalents in Dutch, German, Italian, Russian, and Spanish.* International Council on Archives, Handbook No. 7. 2nd ed., 1988.