COMMON SPECIFICATION FOR INFORMATION PACKAGES

Version: 0.17

AUTHOR(S)	
Name	Organisation
Karin Bredenberg	*National Archives of Sweden*
Björn Skog	*ES Solutions*
Anders Bo Nielsen	*Danish National Archives*
Kathrine Hougaard Edsen Johansen	*Danish National Archives*
Phillip Tømmerholt	*Danish National Archives*
Alex Thirifays	*Danish National Archives // Magenta ApS*
Sven Schlarb	*Austrian Institute of Technology*
Jan Rörden	*Austrian Institute of Technology*
Andrew Wilson	*University of Portsmouth // University of Brighton*
Tarvo Kärberg	*National Archives of Estonia*
Kuldar Aas	*National Archives of Estonia*
Luis Faria	*Keep Solutions*
Helder Silva	*Keep Solutions*
Miguel Ferreira	*Keep Solutions*
Bruno Ferreira	*Keep Solutions*

REVISION HISTORY

Revision No.	Date	Authors(s)	Organisation	Description
0.1	17.02.2014	Björn Skog	ESS	First version.
0.2	21.02.2014	Karin Bredenberg	ESS	Updating content.
0.3	24.02.2014	Björn Skog	ESS	Updating content.
0.4	24.10.2014	Tarvo Kärberg	NAE	Updating content.
0.41	05.11.2014	Tarvo Kärberg	NAE	Adding content from Anders Bo Nielsen.
0.42	08.12.2014	Tarvo Kärberg	NAE	Updating content.
0.43	19.12.2014	Tarvo Kärberg	NAE	Updating content.
0.5	26.01.2015	Kathrine Hougaard Edsen	DNA	Updating content.
0.6	11.02.2015	Tarvo Kärberg	NAE	Rearranging content.
0.7	31.05.2015	Kathrine Hougaard Edsen	DNA	Significant changes suggested
0.8	27.07.2015	Tarvo Kärberg	NAE	Merging content
0.9	05.08.2015	Andrew Wilson	UPHEC	Updating content
0.10	07.10.2015	Kuldar Aas	NAE	Major update to include additional details
0.11	30.11.2015	Kuldar Aas	NAE	Intermediate update to include outcomes and decisions from Common Specification meetings
0.12	08.12.2015	Kuldar Aas	NAE	Update on the implementation, include comments from Sven, Jan (AIT) and Andrew (UPHEC)
0.13	05.01.2016	Kuldar Aas, all	NAE, all	Update to include additional comments from E-ARK WPLs and Common Specification group members version sent for external review
0.14	04.07.2016	Kuldar Aas	NAE	Updated structure → basis for addressing review comments and required updates
0.15	26.08.2016	Kuldar Aas	NAE	Adding available contributions to individual chapters
0.16	05.12.2016	Andrew Wilson, Kuldar Aas	UoB NAE	Major update. Added section on PREMIS. Revision of tables describing use of METS. General text revisions arising from CS meetings. Updates to requirements.
0.17	16.12.2016	All	All	Final discussions, changes and proofreading before delivering the CS to public comment.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENT

The Common Specification for Information Packages was first developed within the E-ARK project in 2014 – 2017. E-ARK was an EC-funded pilot action project in the Competiveness and Innovation Programme 2007-2013, Grant Agreement no. 620998 under the Policy Support Programme.

We would like to thank the National Archives of Sweden and Karin Bredenberg for their support and the availability of the Swedish national Common Specification, upon which most of this document has initially been built.

The authors of this deliverable would like to thank all national archives, tool developers and other stakeholders who provided valuable knowledge about common requirements for information packages and feedback to the this specification!

CONTACT AND FEEDBACK

The Common Specification for Information Packages is maintained by the DLM Archival Standards Board (DAS Board). For further information about the DAS Board or feedback on the current document please consult the website (www.dasboard.eu, **available starting from 1st of February 2017**) or contact us on das@eark-project.eu (**current address is temporary and will be replaced on 1st of February 2017**).

1. Introduction

This document introduces the concept of a Common Specification for Information Packages. It aims to serve three main purposes:

1. Establish a common understanding of the requirements which need to be met in order to achieve interoperability of Information Packages;
2. Establish a common base for the development of more specific Information Package definitions and tools within the digital preservation community;
3. Propose the details of an XML-based implementation of the requirements using, to the largest possible extent, standards which are widely used in international digital preservation.

Ultimately the goal of the Common Specification is to reach a level of interoperability between all Information Packages so that tools implementing the Common Specification can be taken up by institutions without needing further modifications or adaptations.

1.1. The Common Specification and OAIS

In the OAIS[1] framework three types of Information Packages (IPs) are present in a digital preservation ecosystem: Submission Information Packages (SIPs), Archival Information Packages (AIPs) and Dissemination Information Packages (DIPs) (see Figure 1). These three IP types are respectively used to submit data and metadata to digital repositories; store it in long-term preservation facilities; and deliver to consumers.

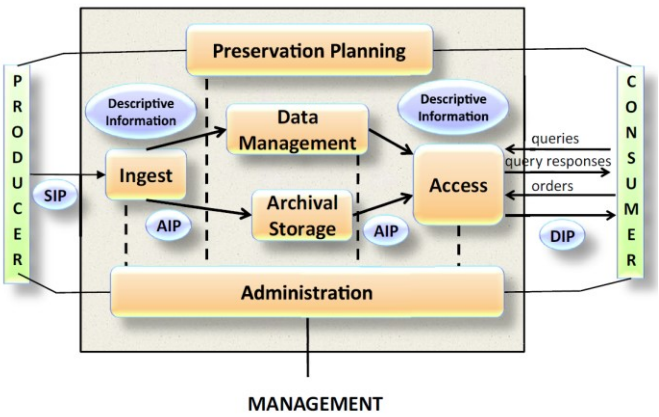


Figure 1: OAIS Functional Entities and Information Packages

This Common Specification aims to summarise the common aspects of all these IPs. The main goal in the development of this specification has been to identify and standardise the common aspects of IPs which are equally relevant and implemented by any of the functional entities of the overall digital preservation process (i.e. pre-ingest, ingest, long-term preservation and access). The practical implementation is that the

[1] Open Archival Information System (OAIS) Reference Model (ISO14721). See: https://public.ccsds.org/Pubs/650x0m2.pdf

specification therefore allows for the development of generic tools and code libraries which can either be applied commonly across the whole lifecycle of digital data, or be reused as the basis for developing more specific, content or process-aware tools.

However, to allow for interoperability on process level there is still a need for defining more detailed technical specifications for a SIP, AIP and DIP. This is also the case for the current Common Specification where more detailed E-ARK SIP, E-ARK AIP and E-ARK DIP profiles[2] have been created.

Figure 2: The scope of Common Specification in regard to OAIS Information Packages.

In general, the E-ARK SIP and E-ARK DIP specifications reuse and apply fully all the requirements set in this Common Specification. However, they also extend it with aspects relevant only for the respective processes (Figure 2).

For example, the E-ARK SIP specification extends the Common Specification with further requirements about recording relevant information on a submission agreement and the actors of the submission process. On the other hand, the E-ARK DIP provides possibilities for describing complex access environments needed to reuse the content of a DIP.

Regarding the E-ARK AIP format, it is important to note that it does not extend the Common Specification in the same way the E-ARK SIP and E-ARK DIP formats do, i.e. in the sense of a format specification inheriting all general properties from the Common Specification (CS) which is then augmented by specific AIP requirements. The reason for this is that while the SIP and the DIP are like "snapshots" in time – one capturing the state of an information package at time of submission (SIP), the other one capturing one form of delivery of the information for access (DIP) – then the AIP needs to deal with an “evolving object” which is constantly updated by preservation actions undertaken in the course of the objects life-cycle. As such, while the E-ARK AIP specification does implement all of the core metadata requirements defined in the Common Specification and extends these (for example it describes a means to record preservation actions about the IP), it does also extend the default structure of the Common Specification (defined in Chapter 4 below). Essentially the AIP introduces a more complex structure which allows at the same time to securely

[2] The relevant E-ARK deliverables (D3.3 for SIP, D4.3 for AIP and D5.3 for DIP) are currently available at http://www.eark-project.com/resources/project-deliverables. Final versions of the IP specifications will be published in February 2017 along with a final version of this Common Specification and published on the DAS Board website (www.dasboard.eu).

hold an E-ARK SIP (which itself follows in full the CS) and at the same time add and modify additional representations over a series of preservation actions.

1.2. Common Specification and Content Information Type Specifications

As an interoperability standard, it must be possible to use the Common Specification regardless of the type and format of the content users need to handle. At the same time, each individual content type and file format can have specific characteristics which need to be taken into account for purposes of validation, preservation and curation.

To allow for such in-depth control over specific content types and formats, the Common Specification introduces the concept of ***Content Information Type Specifications***. A Content Information Type Specification can include detailed requirements on how content, metadata, and documentation for specific content types (for example relational databases or geospatial data) have to be handled within a Common Specification Information Package.

For now (February 2017) there are seven Content Information Type Specifications which have been created by the E-ARK project and have been verified for usage within the Common Specification:

- *SMURF SFSB: Semantically Marked-Up Records Format for Simple File-System Based records.* This Content Information Type Specification describes the usage of the Common Specification for the case of simple computer files organised in folder structures; and their description using the EAD encoding standard[3];
- *SMURF ERMS: Semantically Marked-Up Records Format for Electronic Records Management Systems.* This Content Information Type Specification describes the use of the Common Specification for the archiving of records exported from ERMS-type systems. The specification is built on top of SMURF SFSB and extends it with additional metadata requirements for ERMS-derived metadata;
- *GeoVectorGML and GeoRasterGeoTIFF:* These two Content Information Type specifications build upon the SMURF SFSB and add additional structural and metadata requirements for storing geospatial information, respectively in GML[4] and GeoTIFF[5] formats, within a Common Specification Information Package;
- *SIARD1, SIARD2 and SIARDDK:* These three Content Information Type specifications describe the usage of the Common Specification for the archiving, preservation and reuse of relational databases in one of the formats in the SIARD family (Software Independent Archiving of Relational Databases). Note, that SIARD1 and SIARDDK specifications are deemed outdated by the time of writing and are only intended to be used for packaging already available SIARD1 and SIARDDK packages in Common Specification Information Packages. For new occurrences of archiving relational databases the use of the SIARD2 format[6] and according Content Information Specification is recommended.

[3] https://www.loc.gov/ead/
[4] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=32554
[5] http://trac.osgeo.org/geotiff/
[6] http://eark-project.com/resources/specificationdocs/32-specification-for-siard-format-v20

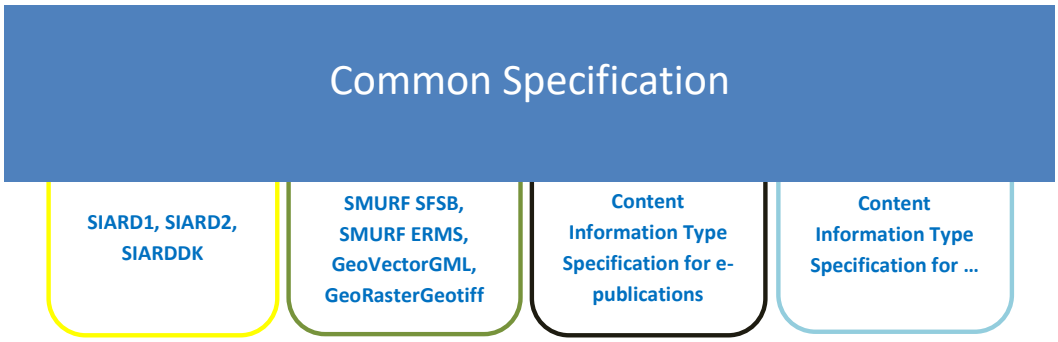


Figure 3: Common Specification and Content Information Type Specifications

The total number of Content Information Type specifications is, however, unlimited and the long-term commitment of the DAS Board[7] is to keep the overall environment open and inclusive. As such, interested bodies are welcome to develop their own Content Information Type Specifications, for example for 3D building projects or electronic publications. An appropriate management regime to facilitate the creation and approval of additional Content Information Type specifications by anyone in the broader community is implemented by the DAS Board (***further information will be available at www.dasboard.eu from 01.02.2017***).

For more detailed information about the Content Information Type specifications please look also at Chapter 6.1 below!

1.3. Common Specification, OAIS Information Packages' specifications and Content Information Type Specifications

Following the discussions in the previous two chapters we can state that the overall ecosystem of the Common Specification consists of 3-layers (see Figure 4):

- The current document, the Common Specification, is the core which provides guidance which must be followed regardless of the process, data or lifecycle stage;
- The E-ARK SIP, AIP and DIP build on the Common Specification and extend it with specific process-related aspects;
- The Content Information Type Specifications define detailed requirements for embedding and describing specific content types within a Common Specification Information Package.

[7] The DLM Archival Standards Board (the DAS Board) is the body committed to maintaing this Common Specification and all related specifications. For further information please consult www.dasboard.eu.

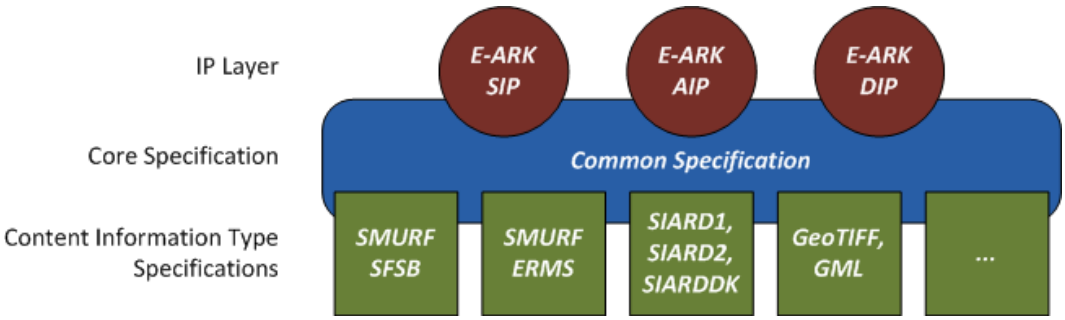


Figure 4: Relations between the Common Specification; E-ARK SIP, AIP and DIP specifications; and Content Information Type Specifications

Therefore the "thing encountered in the wild" is the E-ARK SIP, AIP or DIP including data according to one or many Content Information Type Specifications.

1.4. Relation to other documents

This Common Specification is related to the following documents:

- International standards and best-practices
  - *Reference Model for an Open Archival Information System (OAIS), 2012,* public.ccsds.org/publications/archive/650x0m2.pdf

    This specification has used the same terminology as introduced in the OAIS model and also the same division of information package types: Submission Information Package (SIP), Archival Information Package (AIP), Dissemination Information package (DIP).

  - *Producer-Archive Interface Specification (PAIS) – CCSDS, 2014,* public.ccsds.org/publications/archive/651x1b1.pdf

    We have investigated the structure of a SIP presented in PAIS, but as the implementation of this specification is not very comprehensive yet (only few prototypes exist), we decided to rely mainly on the best practices introduced in other reports (see below).

- E-ARK deliverables[8]
  - Deliverable D3.1, E-ARK Report on Available Best Practices
  - Deliverable D4.1, Report on available formats and restrictions
  - Deliverable D5.1, GAP report between requirements for access and current access solutions

    These three deliverables document the best-practice survey carried out during the first six months of the E-ARK project. Many of the core principles and requirements highlighted in the following chapters have been derived from this survey.

  - Deliverable D3.3, E-ARK SIP Pilot Specification
  - Deliverable D4.3, E-ARK AIP Pilot Specification

[8] All E-ARK deliverables are available at http://www.eark-project.com/resources/project-deliverables

- Deliverable D5.3, E-ARK DIP Pilot Specification

  The E-ARK SIP, AIP and DIP specifications build on the Common Specification and extend it in regard to requirements derived from pre-ingest and ingest, archival storage, and access processes.

1.5. Structure of the document

The rest of this document describes the Common Specification and its practical implementation. The document is divided into two logical parts.

The first part (Chapters 2 and 3) describes the generic principles of a Common Specification for Information Packages. The main aim of these chapters is to first identify a common set of needs and thereafter present a series of requirements which an Information Package needs to follow regardless of the implementation at any given point in time:

- **Chapter 2** provides an explanation of the need for a Common Specification for Information Packages. The chapter therefore presents some practical use cases which highlight the potential savings and increased functionality of digital archives when following internationally standardised approaches.
- **Chapter 3** presents the core requirements which need to be met in order to achieve the interoperability goal described in Chapter 2. Based on these requirements a set of high-level solutions are introduced regarding, for example, the structure and use of metadata within any implementation of an Information Package.

The second part of this document (Chapters 4, 5 and 6) presents a practical implementation of the principles described in previous chapters, as implemented according to current state-of-the-art technologies. As such, this part of the document describes the requirements which are needed to achieve practical IP interoperability:

- **Chapter 4** presents a detailed description of the structure which must be implemented in any Common Specification Information Packages
- **Chapter 5** presents a detailed overview of metadata requirements within Common Specification Information Packages with a special focus on the use of metadata elements which are needed for the automation and interoperability of archival validation and identification tasks
- **Chapter 6 (to be finalised by 01.02.2017)** describes additional (optional) components extending the practical implementation in regard to specific aspects
  - How to create new Content Information Type specifications
  - How to split large content objects between multiple physical IPs
  - Guidelines on adding (any) descriptive metadata into a Common Specification Information Package
  - Technical requirements for building validators for Common Specification Information Packages.

Finally, in addition to this document full examples of IPs conforming to the Common Specification implementation details are available at https://github.com/eark-project/information-package

PART I: Common Specification for Information Packages

In this part of the document we build the argument for a Common Specification for Information Packages and present the main concepts and requirements for the purpose.

2. Need for establishing common ground

> ***The vision:*** *All digital preservation systems receive, store and provide access to information, regardless of its size, type or format, according to a set of agreed principles which allow institutions to identify, verify and validate the information in a uniform way.*
>
> ***The goal:*** *Interoperability between data sources, archives and reuse environments is improved to a point where digital preservation tools can be reused across borders and institutions. This opens up new possibilities for collaboration and limits greatly the need for development resources for any single institution.*

The amount of digital information being created, held and exchanged is continuously growing. This information is created with the help of numerous software tools and systems, comes in a variety of technical formats, and covers most aspects of our daily lives. Regardless of the formats and systems in question we always need to consider whether the information is needed to be retained and managed for longer periods of time. The reasons for this might be, for example:

- to meet legal and regulatory obligations
- to provide for efficient reuse
- to satisfy historical, cultural, scientific and business interest.

As of now, most tools and systems used to create information are not built for coping with long-term requirements of keeping information safe and accessible. Instead, implementations separate the short-term and long-term management of information into different systems, for example business and records systems on one hand and archival systems on the other (Figure 5).

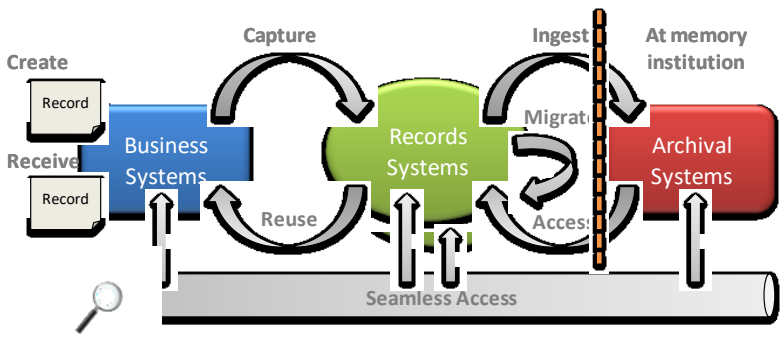


Figure 5: Information flow between live and archival systems

The implication for data owners and system managers is that information which has to be kept for extended time periods needs to be exchanged between a set of different locations, including archival systems:

- as effectively as possible,
- without endangering the authenticity and integrity of the information,
- and without limiting the possibilities for discovering and reusing the information.

As such, what we need in order to make the long-term availability of crucial information possible under (usually limited) resources is a set of principles which allow exchanging information in a common way across the systems participating in archival workflows and processes, i.e. create a set of IP interoperability specifications. For the Common Specification we have identified the following interoperability scenarios (Figure 6):

- Export of data and metadata from source systems and transfer to SIP creation tools (or directly, as an SIP, into preservation systems);
- Transfer of SIPs between SIP creation tools and preservation systems;
- Exchange of preservation system platform where all AIPs need to be migrated into a new technological platform;
- Distributed storage and synchronisation of AIPs between multiple (technologically different) preservation systems;
- Exchange of DIPs between preservation systems and access platforms or portals;
- Exchange of DIPs between various access platforms of portals.

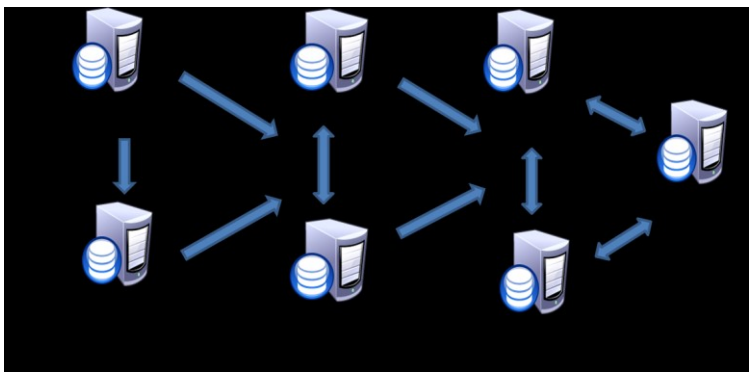


Figure 6: Archival workflow and tool ecosystem

As of 2014 (the start of the E-ARK project) the state of interoperability in digital preservation was rather poor. While national or institutional practical implementation-level specifications existed to serve the need for IP packaging and exchange, these were by large not interoperable with each other. On the contrary, available and widely used international specifications (most notably METS[9] and PREMIS[10]) lack the necessary implementation-level detail, needed in order to serve as an authorative source for practical interoperability.

[9] http://www.loc.gov/standards/mets/
[10] http://www.loc.gov/standards/premis/

This situation has a remarkable effect on the cost of digital preservation. Namely, the tools developed in individual institutions are not reusable across institutional and state borders and therefore need to be redeveloped at each single location. Globally, this raises the cost of digital preservation to a level which makes it not affordable for smaller institutions and, at the same time, does often not allow developing tools which would be sufficiently mature, user-friendly and prone to errors. As well, the multitude of national or institutional specifications does not allow internationally active source system providers (e.g. Oracle, Microsoft) to build a single native archiving functionality into their products, meaning that there is a need for bespoke development (and therefore added cost) for each installation of these source systems across all sectors and countries.

To overcome these limitations this document proposes a universal common specification, which can be implemented across borders, for how data and metadata should be structured and packaged when transferred to archival systems, ingested and preserved in these, and re-used. Such a specification will allow data owners to build standardised interfaces for the export of their data regardless of the archives in question; and digital archives to build standardised interfaces for data ingest and access, regardless of the data providers and users in question.

Further, the aim of the common specification is to be sufficiently detailed and technical to allow for extended collaboration in regard to software development and pooling. Ideally the tools which implement the common specification for data export, transfer, ingest, preservation and reuse are exchangeable between institutions and administrations with minimal effort. This in turn shall lead to a significant decrease in resources needed from any single institution and at the same time opens up an extended market for commercial software providers.

3. Requirements for Common Specification Information Packages

At the heart of any standardisation activity is achieving a common understanding of what needs to be standardised and for what purposes. This is also the goal of this chapter, which presents a series of high level requirements for an Information Package. Most of the requirements are driven by the need for interoperability – the Information Packages built according to the requirements need to be easy to exchange, identify, validate and (re)use.

Another crucial factor to take into account is long-term sustainability. Practical technical and semantic interoperability is possible only when a certain set of technologies have been agreed upon and implemented. However, especially in the field of digital preservation, any technology will become outdated sooner or later and agreed approaches will need to be updated to accommodate new, better and more efficient technologies and standards. Because of this, the developers of this Common Specification have reused, as much as possible, existing powerful, standardised and well-established best practices for the technical implementation of an Information Package (see Part II of this document). This does not mean that the technical implementation will not need to be changed, only that the need will arise later rather than sooner. So, to achieve long-term sustainability of the Common Specification, we present below a set of generic requirements which must be followed when updating any of the technologies used in a technical implementation at any given point in time.

Ultimately the requirements below present a conceptual view of an Information Package, including an overall IP data model, and use of data and metadata. An implementation of this conceptual view is presented later, in Part II of this document.

The requirements are described in a straightforward way – each requirement has a sequential number and a short description. The description includes always a MoSCoW (MUST/MUST NOT, SHOULD/SHOULD NOT, COULD, WOULD) prioritisation statement[11]. The short description of each requirement is followed by a rationale which describes the reason and background for the requirement.

3.1. General requirements

Requirement 1.1: *It MUST be possible to include any data or metadata, regardless of its type or format, in a Common Specification Information Package.*

This is one of the most crucial requirements of the Common Specification. In order to be truly "common" technical implementations of the Common Specification MUST NOT introduce limitations or restrictions which are only applicable to certain data or metadata types. If an Information Package definition fails to meet this requirement it is not possible to use it across different sectors and tools, thereby limiting practical interoperability.

Requirement 1.2: *A Common Specification Information Package MUST NOT restrict the means, methods or tools for exchanging it.*

[11] For more information on the MoSCoW method see for example: https://en.wikipedia.org/wiki/MoSCoW_method

Tools and methods for transferring Information Packages between locations are constantly evolving. It is also possible that different methods might be preferred especially for packages of varying sizes. In order to achieve that a Common Specification Information Package is truly interoperable across different platforms it therefore MUST NOT introduce limitations or restrictions which would be impossible to be met by specific information exchange tools or channels.

As such the Common Specification does also not define the requirement to use a particular transfer package or envelope. The scope of the Common Specification is limited to the structure and requirements for data and metadata within the package. Different implementers are welcome to choose their own methods on top of the Common Specification.

Requirement 1.3: *The Common Specification MUST NOT define the scope of data and metadata which constitutes an Information Package.*

One of the fundamental principles of the Common Specification is that it MUST allow each individual repository to define the (intellectual) scope of an Information Package and its relations to real life entities. As such, any implementation of the Common Specification MUST be equally usable for packaging the content of an whole information system (for example an ERMS) as an single IP; or when extracting only one record and its metadata from the information system and packaging as an single IP (or anything between these two extremes).

Out of the previous we can also derive that a Common Specification MUST NOT define whether, for example, a SIP should conform to exactly one or many AIPs. Instead the Common Specification MUST allow for the inclusion of "anything that the implementer wants to define as a SIP, AIP or DIP" and allow for "any (1-1; 1-n; n-1; n-m) relationships between SIPs, AIPs and DIPs.

Requirement 1.4: *A Common Specification Information Package SHOULD be highly scalable.*

One of the practical concerns for Information Packages is their size. Many digital repositories have problems with data objects and metadata of increasing sizes, making it especially difficult to carry out tasks related to data or metadata validation, and identification and modification.

Consequently, it is our recommendation to provide for appropriate scalability mechanisms (for example: mechanisms for splitting large-scale data or metadata) when devising any implementation for the Common Specification.

Requirement 1.5: *A Common Specification Information Package MUST be machine-readable*

To support the goal of automating ingest, preservation and access workflows each of the implementations of the Common Specification must be machine-actionable. This means that decisions about the use of metadata syntax and semantics as well as the physical structure must be expressed explicitly and in a clear way. This, in turn, allows the specification to be implemented in the same way across different tools and environments.

Requirement 1.6: *A Common Specification Information Package SHOULD be human-readable*

In long-term preservation we also need to take into account that "forgotten" Information Packages might be found long after details about the implementation are gone and no tools to access the package are

available. For these scenarios it is crucial to ensure that the structure and metadata of the Information Package are understandable with minimal effort by using simple tools like text editors and file viewers.

In practice this means that any implementation of the Common Specification should ensure that folder and file naming conventions allow for the human identification of package components, and that the semantics of the package is explicit.

Requirement 1.7: *A Common Specification Information Package MUST support the preservation method best suited for the data.*

Different preservation institutions and different types of data need to use different methods for long-term preservation; migration and emulation being the most usual choices. A Common Specification Information Package MUST NOT prescribe the use of a specific preservation method but instead allow to document and/or add any data or metadata which is needed for any method.

3.2. Identification of the Information Package

Requirement 2.1: *The Information Package type (SIP, AIP or DIP) MUST be clearly indicated.*

One of the first tasks in analysing any Information Package is to identify its current status in the overall archival process. Therefore, any Information Package must explicitly and uniformly include metadata which identifies it as a SIP, AIP or DIP.

Requirement 2.2: *The Information Package MUST clearly indicate the Content Information Type(s) of its data and metadata.*

As stated in *Requirement 1.1* a Common Specification Information Package must be able to include any kind of data and metadata. At the same time we have introduced in earlier chapters the concept of Content Information Types which allow users to achieve more detailed control and fine-grained interoperability. As such, any Common Specification Information Package MUST include a statement about which Content Information Type specification(s) has been followed within the Information Package, or on the contrary, indicate clearly that no specific Content Information Type Specification has been followed.

The practical implication of requirements 1.1, 2.1 and 2.2 is that if these have been followed we can in fact develop modular identification and validation tools and workflows. While generic components can carry out high level tasks regardless of the Content Information Type, it is possible to detect automatically which additional content-aware modules need to be executed.

Requirement 2.3: *A Common Specification Information Package MUST bear an identifier which is unique and persistent in the scope of the repository.*

In order to manage a digital repository and provide access services each Information Package stored in the repository MUST be identified uniquely at least within the repository. At the same time a Common Specification implementation MUST NOT limit the choice of the exact identification mechanism, as long as the mechanism is implemented consistently throughout the repository.

Requirement 2.4: *A Common Specification Information Package SHOULD bear an identifier which is globally unique and persistent.*

In addition to the previous requirement, it is recommended that the identification mechanism used at the repository provides for global uniqueness and persistence of Information Package IDs. The application of globally unique and persistent identifiers allows repositories to participate more easily in cross-institutional information exchange and reuse scenarios (for example participation in national or international portals, or cross-repository duplication of AIP preservation). However, the Common Specification MUST NOT limit the choice of the exact identification mechanism.

Requirement 2.5: *All components of a Common Specification Information Package MUST bear an identifier which is unique and persistent within the repository.*

As stated above, a Common Specification Information Package MUST be flexible enough to allow for the inclusion of any data or metadata depending on the needs of the repository and its users. As well, an Information Package might include additional support documentation like metadata schemas, user guidelines, contextual documentation etc. Regardless of which and how many components constitute a full Information Package, all components MUST bear a unique and persistent identifier which allows for the appropriate linking of data, metadata and all other components. This, in turn, is one of the most crucial aspects towards achieving an interoperable way towards maintaining package integrity.

It is also worth mentioning that in any implementation it is only necessary to achieve identifier uniqueness and persistence within an individual Information Package. If this is the case, repository-wide uniqueness is easily achieved when combining the package ID (unique according to requirement 2.3) and the component ID.

Note: The components of a Common Specification Information Package are explained in more detail in chapter 3.3.

3.3. Structure of the Information Package

Requirement 3.1: *A Common Specification Information Package MUST be built in such a way that its data and metadata can be logically and physically separated from one another.*

At the highest level each Information Package can be divided into data and metadata. In order to minimise the effort needed for the identification and validation of both, and to simplify long-term preservation actions it is reasonable to clearly separate data and metadata. This allows, for example, ingest tools to streamline and separate metadata identification and validation tasks, and file format identification and normalisation. Throughout long-term preservation such a separation allows also to update respective data or metadata portions of an Information Package without endangering the integrity of the whole package.

Requirement 3.2: *The structure of the Information Package SHOULD allow for the separation of different types of metadata*

In addition to the previous requirement it is recommended to explicitly divide metadata into more specific components. While the definitions of metadata types vary a lot between implementations it is our

recommendation to divide metadata logically and physically at least into descriptive and preservation metadata.

Requirement 3.3: *The structure of the Information Package MUST allow for the separation of data and metadata representations.*

The concept of representations is one of the fundamental building blocks in digital preservation. As technologies evolve and get obsolete, data and metadata is constantly updated in order to ensure long-term accessibility, therefore creating new versions or representations of the data and metadata.

Expressing representations within the logical and physical structure of an Information Package helps institutions to explicitly understand the various states of the information throughout its lifecycle, therefore improving also the ease of long-term management and reuse of the information.

Requirement 3.4: *The structure of a Common Specification Information Package MUST explicitly define the possibilities for adding additional logical components into the Information Package.*

In addition to data and metadata, institutions might have the need to include additional information in an Information Package. For example, implementers might decide that XML Schemas about metadata structures and additional binary documentation about the original IT environment have to be added to the package.

If this is the case, the Common Specification Information Package MUST NOT limit which components can constitute an Information Package, and MUST offer clearly defined extension points for the inclusion of these additional components into the Information Package. At the same time these extension points MUST be defined in a way which does not interfere with other components (i.e. the extension points MUST be clearly separated from other components of an Information Package).

Requirement 3.5: *A Common Specification Information Package MUST follow a common conceptual structure regardless of its technical implementation.*

Based on requirements 3.1 – 3.4 we now present a common structure for any Common Specification Information Package (Figure 7).

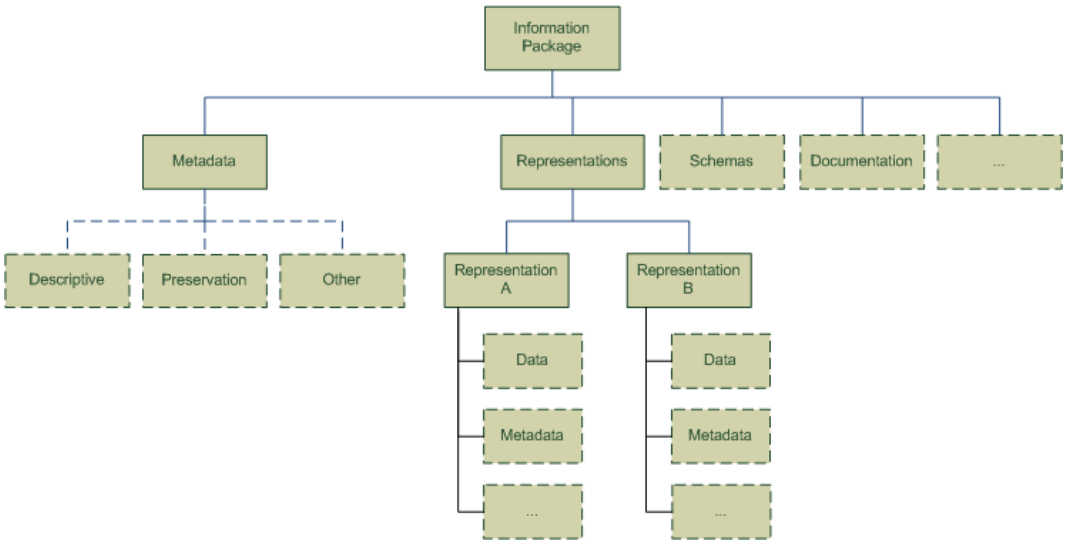


Figure 7: Conceptual structure of the Common Specification

Following *Requirement 3.3* the structure separates explicitly the representations of data and metadata into a separate structural component.

Following *Requirement 3.1* the package MUST include a high-level structural component for metadata which includes at least relevant metadata for the whole package. In addition the representations MUST internally separate between data and metadata (though note that the Common Specification does not mandate that both data and metadata must be available in all representations).

In addition we highly recommend dividing the metadata portion of the Information Package to separate different types of metadata *(SHOULD Requirement 3.2).*

Following *Requirement 3.4* repositories and their users have the possibility to add any additional components (as an example for schemas and binary support documentation) either as extensions to the whole Information Package or into a specific representation.

This common structure MUST be followed throughout any specific physical implementation of the Common Specification.

Requirement 3.6: *A Common Specification Information Package MUST be implemented by one and only one implementation at any point in time.*

The conceptual structure presented above can be implemented in various ways – for example the components might be defined by accompanying package metadata or explicitly through a physical structure. However, it is not reasonable to have multiple (competing) implementations available at once as this would lead to unnecessary complexity in developing tools needed for creating, processing and managing Information Packages.

At the same time it is clear that any given technical implementation will become obsolete in time, for example as new transfer methods and storage solutions emerge. As such this requirement does not prohibit the take-up of any emerging logical of physical technical solutions but merely requires to have one and only one of these to be implemented at any given point in time.

At the time being, the Common Specification mandates a fixed physical folder structure (see Chapter **Tõrge! Ei leia viiteallikat.**) as the implementation of this and the previous requirement.

3.4. Information Package Metadata

Requirement 4.1: *Metadata in a Common Specification Information Package MUST be based on standards.*

In order to exchange, validate, process and reuse Information Packages in an interoperable and automated way we need to standardise how crucial metadata are presented in the package. “Crucial metadata”, we see mainly as the core information about how the package content has been created and managed (administrative and preservation metadata), explicit descriptions about of the structure of the package (structural metadata) and the technical details of the data themselves (technical metadata).

In order to ensure that these metadata are understood and implemented in a common and interoperable way in any Information Package, the use of established and widely used metadata standards is highly recommended.

In the current implementation a large proportion of such metadata is covered by the widely used METS and PREMIS standards (see Chapter 5).

Requirement 4.2: *Metadata in a Common Specification Information Package MUST allow for unambiguous use.*

Many metadata standards support multiple options for describing specific details of an Information Package. However, such interpretation possibilities can also lead to different implementations and ultimately to the loss of interoperability.

To overcome this risk the Common Specification requires that, while developing a specific implementation, the chosen metadata standard MUST be reviewed in regard to potential ambiguity. If needed, the selected metadata standard MUST be further refined to meet the needs of interoperability and automation.

Requirement 4.3: *A Common Specification Information Package MUST NOT restrict the addition of any additional metadata.*

Previous requirements state the importance of highly controlled administrative, preservation, structural and technical metadata for interoperability purposes. At the same time the opposite applies for other types of metadata, most prominently for resource discovery (also called descriptive) or Content Information Type specific technical and structural metadata. In order to not limit the widespread adoption of this Common Specification it has to be possible for any implementer to add any metadata next to the mandatory metadata components needed for package level automation and interoperability.

In case organisations need to prescribe further details about descriptive or Content Information Type specific metadata for a deeper level of interoperability it is possible to use the mechanism of Content Information Type Specifications described above.

To summarise the requirements above from a more technical perspective, the Common Specification foresees a modular approach towards Information Package metadata:

- All Information Packages share a common core of metadata which allows for the common development of high-level package creation, validation, identification and reuse tools;
- The rest of the metadata in the Information Package might follow additional agreements which have been made in order to develop specific tools such as, for example, tools to manage archival descriptions in EAD, or for specific Content Information Types like relational databases in the SIARD2 format.

PART II: Implementation of the Common Specification

In this part of the document we present an implementation of the requirements and principles discussed in Part I of this Common Specification. The implementation consists of two core elements: a fixed physical structure of a Common Specification Information Package (Chapter 4) and the exact use of metadata in METS and PREMIS format (Chapter 5).

As explained above, any implementation is destined to be outdated sooner or later. However, the creators of the Common Specification have made their best effort to reuse already available best practices and established core standards, and to carry out intensive discussions within the digital preservation community. All of the above should guarantee that the implementation can be used with only minor updates (for example minor updates to metadata elements) throughout the next few decades.

4. Common Specification Information Package structure

The implementation of the conceptual model described in Requirement 3.5 is a fixed physical (folder) structure which follows exactly the components conceptual structure.

The main reason for such an implementation decision is that a fixed physical folder structure makes it clear for both human users and tools where to find what. The main benefit of such a clear decision is that many archival tasks (for example file format risk analysis) can be executed directly on the data portion of the package structure, as opposed to first processing potentially large amounts of metadata for the locations of the files. This, in turn, allows for more efficient processing which is valuable in the case of large collections and bulk operations. In short, we believe that a fixed folder structure allows for more efficiency and scalability.

The authors of this specification are well aware that there are multiple data storage solutions which do not support explicit folder structures but use other means for structuring and storing (the content of) AIPs. However, we would like to note that the purpose of this specification is to support Information Package interoperability. As such we believe that even if a storage solution does not allow implementing the physical folder structure as the native AIP storage structure, it is still possible to implement the physical structure described below for SIPs, DIPs and the import/export of AIPs. While the repository needs to support an extra transformation (i.e. Common Specification IP to internal AIP and vice versa), it allows still to use the tools created by other users of the common specification, transfer AIPs more easily to new repository systems or storage solutions, and establish cross-repository duplicated storage solutions.

4.1. Folder structure of the Common Specification Information Package

The Common Specification Information Package folder structure is presented in Figure 8 below. The structure follows directly the principles of the conceptual data model by dividing the components of the package into stand-alone folders for representations, metadata, and other components.

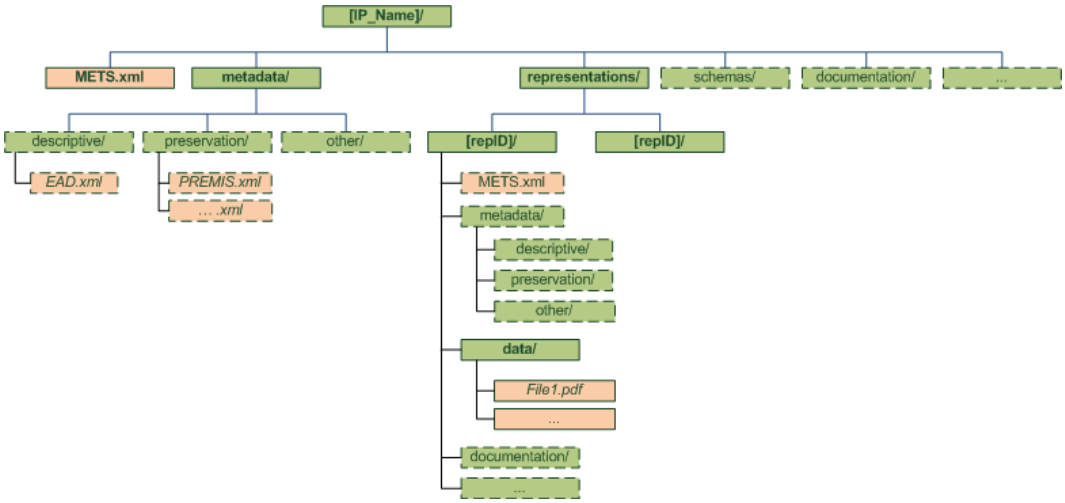


Figure 8: Common Specification Information Package folder structure

The implementation requirements of the Common Specification Information Package structure are:

- Each Common Specification Information Package MUST be included in a single physical folder (i.e. the "Information Package folder"). In other words: on the highest structural level a Common Specification IP MUST consist of one and only one folder;
- The Information Package folder SHOULD be named with the ID or name of the Information Package;
- The Information Package folder CAN be compressed (for example by using TAR or ZIP);
- The Information Package folder MUST include a metadata file named "**METS.xml**", which includes information about the identity and structure of the package and its components[12];
- The Information Package folder MUST include a folder named "**metadata**", which MUST include at least all metadata relevant for the whole package[13]
    - If preservation metadata are available, they SHOULD be included in sub-folder "**preservation"**;
    - If descriptive metadata are available, they SHOULD be included in sub-folder **"descriptive"**;
    - If any other metadata are available, they CAN be included in separate sub-folders, for example an additional folder named "**other**".

[12] For a detailed description of the content of the METS.xml file please consult chapter 5.
[13] As a convention the "**metadata**" folder MUST be present even if there is no metadata within the package (for example for a SIP).

- The Information Package folder MUST include a folder named **"representations"**;
    - The "**representations**" folder MUST include a sub-folder for each individual representation (i.e. the "representation folder") named with a string uniquely identifying the representation within the scope of the package (for example the name of the representation and/or its creation date could be good examples for an representation sub-folder)[14];
    - The representation folder MUST include a sub-folder named "**data**" which includes all data constituting the representation[15];
    - The representation folder CAN include a metadata file named "METS.xml" which includes information about the identity and structure of the representation;
    - The representation folder CAN include a sub-folder named "**metadata**" which CAN include all metadata about the specific representation
- The Information Package folder and representation folder CAN be extended with additional sub-folders:
    - We recommend including XML Schemas for all metadata in XML format into the package. These schemas SHOULD be placed into the sub-folder called "**schemas**" within the Information Package folder;
    - We recommend including all additional (binary) documentation about the whole package or a specific representation into the package. Such documentation SHOULD be placed into the sub-folder called "**documentation"** within the Information Package folder and/or the representation folder;
    - Implementers CAN add any other folders either into the Information Package folder or the representation folder.

4.2. Implementing the structure

The requirements presented in chapter 4.1 leave room for quite a few decisions during implementation. For the sake of clarity we provide here examples for two extremes – the simplest and the full use of the structure.

In the simplest case the structure can be implemented following mostly just the MUST requirements. An example of this is visible on Figure 9.

[14] Note that the structure **does not** require the inclusion of all representations in a single package. If institutions prefer to keep different representations as separate packages they are welcome to do so. However, to allow for consistent tool support the "**representations**" folder MUST be available and the naming followed even if only one representation exists in each individual Information Package.

[15] For the time being the Common Specification supports only full representations (i.e. all data constituting a representation MUST be explicitly available within the „**data**" folder). Support for partial representations (i.e. an intellectual full representation consists of multiple physical representations) is expected to be available by 2018.

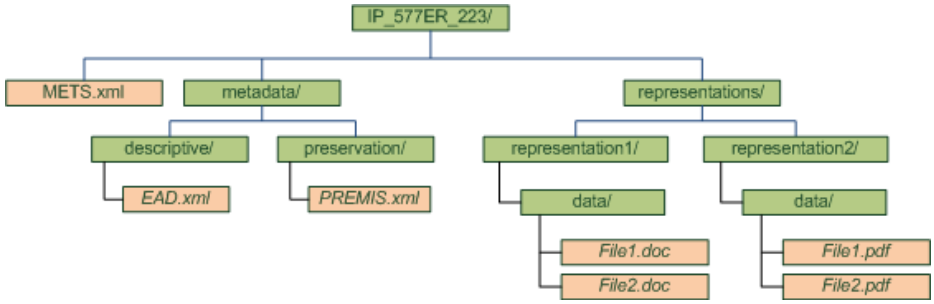


Figure 9: Example of a simple use of the Common Specification structure

The main point to highlight with such a simple use is that the representations have been kept as simple as possible. All metadata about both the package and the representations (in this example METS, EAD and PREMIS metadata) are located in the Information Package folder and none of these components are available within the representation folders.

Such a simple implementation is reasonable in scenarios where the amount of data and metadata is limited. However, in the case of large Information Packages (for example, a package including three representations and 1,000,000 files in one representation) the size of both the METS.xml file and preservation metadata can grow too large to manage efficiently. Especially in such large data scenarios it might prove necessary to implement all the capabilities of the structure presented in the previous chapter.

An example of the full implementation is delivered in Figure 10. The main difference between the simple and full use of the structure is that each representation does essentially repeat the simple structure. Especially structural and preservation metadata in METS and PREMIS formats is available in both the Information Package folder (for package level descriptions) and within representation folders (for representation level descriptions). As such the full structure allows for easier management of single representations and brings further benefits like more straight-forward metadata versioning.

It is worth to note that, in order to avoid confusion, it is recommended to have a common approach towards adding metadata into representations or not. In other words, we recommend having all representation-relevant metadata either in the root metadata folder or the representation metadata folder, but not to have a mixed approach (i.e. some representation metadata in the root metadata folder and some within the representation). Further, we do not recommend the duplication of any metadata or the content of optional folders (schemas, documentation, etc.) between the Information Package folder and representation folders.

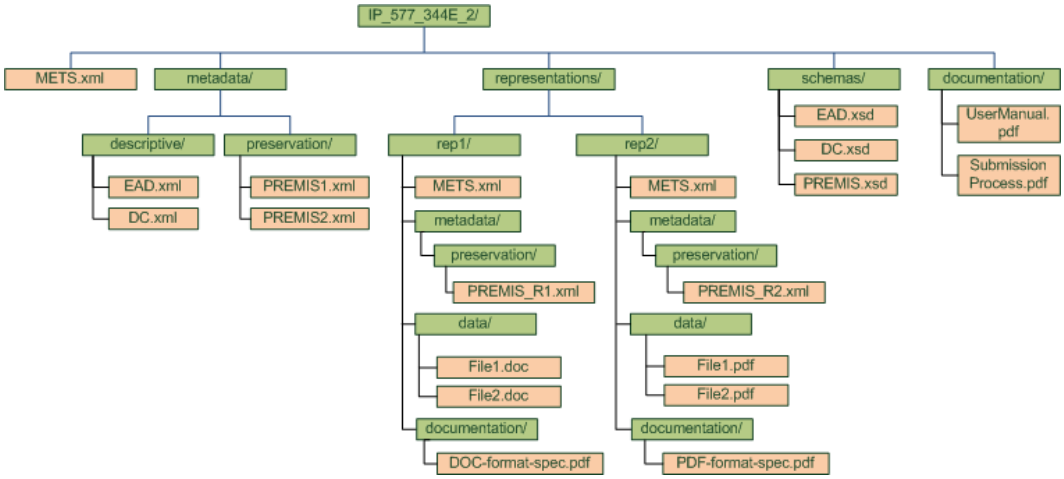


Figure 10: Example of the full use of the Common Specification structure

5. Use of metadata

5.1. General requirements for metadata in a Common Specification Information Package

The number one consideration when discussing metadata requirements is, as with the rest of this specification, the need for interoperability. In more detail, the focus is on high-level technical interoperability and tasks which allow an Information Package to be prepared, transferred and received regardless of the institutions and tools involved. These tasks include:

- Identifying uniquely an Information Package and its components;
- Validating an Information Package;
- Validating the contents of an Information Package;
- Proving the authenticity of the Information Package;
- Accessing the contents of an Information Package.

In more technical terms this Common Specification makes an effort to control metadata which allows any tool or user to negotiate the data and metadata components of the package (i.e. packaging metadata), to validate that no component has come to harm during transfer or preservation (i.e. fixity information), to understand the processes behind the creation and management of the package (i.e. provenance and preservation metadata) and finally to understand how the data within the package could be accessed (i.e. representation information).

Most crucially, we regard descriptive metadata and most of detailed technical metadata to not belong in the scope of the Common Specification. As such, the Common Specification itself does not aim to provide detailed semantic interoperability between different systems. However, as noted in chapter 1.2, implementers are welcome to use the construct of Content Information Type Specifications to achieve an even higher level of interoperability.

Some of the core metadata requirements are already visible from the structure presented in the previous chapter. Most crucially the Common Specification requires that all Information Packages MUST include one and only one METS file in the **Information Package folder** of the package, named “METS.xml”. In addition, the package CAN include one “METS.xml” file in each of the representation folders. These files will be referred to as “root METS” and “representation METS” respectively in the rest of this document. The detailed specification of using METS within the Common Specification is available in Chapter 5.3.

In addition to the METS files the Common Specification ***recommends*** the inclusion of PREMIS metadata in appropriate preservation metadata folders. This is especially relevant when aiming for an interoperable approach towards provenance and access to Information Packages. However, we recognise that, especially in the case of SIPs, appropriate preservation metadata is not always available. As such this is also not an absolute requirement though highly desirable. The detailed specification of the use of PREMIS within the Common Specification is available in Chapter 5.4.

The use of any additional metadata is not restricted in Common Specification Information Packages.

5.2. General requirements for the use of metadata

Before we describe the detailed requirements for the use of METS and PREMIS we would like to highlight some general aspects which need to be implemented commonly across all metadata.

- **The use of identifiers**

  The ID data type in XML does by default not allow for identifiers which start with a number. To overcome this limitation and in order to allow for interoperable package identification all identifiers within Common Specification metadata MUST start with an identifier prefix, followed by a colon, and the actual value of the identifier.

  Example: `OBJID="UUID:5d378f86-28a1-41d8-a2b9-264b10fbd511"`

- **Referencing between files within a Common Specification IP**

  A common approach towards referencing between metadata, and between metadata and other components of the package, is one of the core needs in Information Package validation and integrity checking. Different technical solutions are available for referencing and not all of these are supported across all digital preservation tools.

  In order to guarantee interoperability all references within a Common Specification Information Package:

  - CAN indicate the protocol part ("file://"), in which case the path MUST be expressed a valid URI according to RFC 3986[16];
  - If the protocol part is omitted, the path MUST be interpreted as a relative reference to the metadata file from which the reference originates.

  Example: `xlink:href="metadata/descriptive/EAD.xml"`

- **Referencing other packages**

  As with internal referencing it is crucial that external references to other related packages are expressed in an interoperable manner. As such all references to other Common Specification Information Packages MUST use the value of the *mets/@OBJID* attribute of the package.

5.3. Use of METS in a Common Specification Information Package

The main requirement for METS files in a Common Specification Information Package is that these need to follow the official METS Schema version 1.11[17]. As new versions of METS Schema become available the DAS Board will evaluate these and, if necessary, update the Common Specification respectively.

[16] Available at https://tools.ietf.org/html/rfc3986
[17] Available at http://www.loc.gov/standards/mets/version111/mets.xsd

The following text assumes knowledge of the principles of the METS specifications. If this is not the case, please consult the official documentation[18] before continuing.

The rest of this chapter is structured according to the core METS elements: METS root element *mets*, *header*, *amdSec*, *dmdSec*, *fileSec*, *structMap*, and *behaviourSec*. In each of these sections we explain in a concise way limitations imposed by the Common Specification implementation when compared to the official METS documentation. Also, differences between creating a root METS file and representation METS file are described when relevant.

All names of elements and attributes below are expressed using the XLink notation (i.e. *element/sub-element/@attribute*)

5.3.1. Use of the METS root element (element *mets*)

The purpose of the METS root element is to describe the container for the information being stored and/or transmitted, which is held within the seven sections of the METS file. The root element of a METS document has five attributes derived from the official METS specification and one attribute added for the purposes of this Common Specification.

In addition to these six attributes the METS root element *mets* MUST define all relevant namespaces and locations of XML schemas using the *@xmlns* and *@xsi:schemaLocation* attributes. In case XML schemas have been included into the package (i.e. placed into the "**schemas**" folder) it is recommended to link to the schemas using the relative path of the schema file (i.e. *schemas/mets.xsd*).

The specific requirements for the root element and its attributes are described in the following table[19].

Name	*Element/ Attribute*	*Description and usage*	*Cardinality*
METS root element	mets	The root level element that is required in all METS documents	1..1
Root ID	mets/@ID	Optional, no further requirements	0..1
Content ID	mets/@OBJID	Mandatory in this specification. It is recommended that it be the same as the name or ID of the package (the name of the root folder). The OBJID must meet the Common Specification requirement of being unique at least across the repository	1..1
Package name	mets/@LABEL	Optional, if used should be filled with a human-readable description of the package	0..1
General content type	mets/@TYPE	Mandatory in this specification. The TYPE attribute must be used for identifying the type of the package (genre), for example	1..1

[18] Available at http://www.loc.gov/standards/mets/mets-schemadocs.html

[19] Please note that here and in similar tables in next sub-chapters we list only these METS elements which have been further restricted within E-ARK (when compared to the official METS schema documentations). Implementers can use all other METS elements not listed in the tables according to their best practices and the official METS schema documentation.

		ERMS, RDBMS, digitised construction plans. However, there is no fixed vocabulary and as such implementers are welcome to use values most suitable for their needs.	
Content Information Type Specification name	@CONTENTTYPESPECIFICATION	An attribute added by this specification. It describes which content information type specification is used for the content. Values of the attribute are fixed in the following vocabulary: 1. SMURFERMS 2. SMURFSFSB 3. SIARD1 4. SIARD2 5. SIARDDK 6. GeoVectorGML 7. GeoRasterGeotiff NB The vocabulary is extensible as additional content information type specifications are developed.	0..1
METS profile	@PROFILE	Mandatory in this specification. The PROFILE attribute has to have as its value the URL of the official Common Specification METS Profile[20].	1..1

Full example of the METS root element:

```
<mets xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns="http://www.loc.gov/METS/"
PROFILE="http://www.eark-project.com/METS/IP.xml" TYPE="RDBMS" CONTENTTYPESPECIFICATION="SIARD2"
OBJID="UUID:5d378f86-28a1-41d8-a2b9-264b10fbd511" LABEL="METS file describing the AIP matching
the OBJID." xsi:schemaLocation="http://www.loc.gov/METS/ schemas/IP.xsd
http://www.w3.org/1999/xlink schemas/xlink.xsd">
```

5.3.2. Use of the METS header (element *metsHdr*)

The purpose of the METS header section is to describe the METS document itself, for example information about the creator of the IP.

The requirements for the *metsHdr* element, its sub-elements and attributes are presented in the following table.

Name	*Element/ Attribute*	*Description and usage*	*Cardin*

[20] The official METS profile is currently (December 2016) being prepared. Until it is available the placeholder value to be used is "http://www.eark-project.com/METS/IP.xml"

			ality
METS Header	metsHdr	Element for describing the package itself	0..1
METS Header ID	metsHdr/@ID	Optional, no further requirements	0..1
Administrative Metadata ID	metsHdr/@ADMID	Optional, referring to the appropriate administrative metadata section, if used for metadata about the package as a whole.	0..1
Package creation date	metsHdr/@CREATEDATE	Mandatory, the date of creation of the package	1..1
Package last modification date	metsHdr/@LASTMODDATE	Mandatory if relevant (in case the package has been modified)	0..n
Package status	metsHdr/@RECORDSTATUS	Optional, no further requirements	0..1
OAIS Information Package Type	metsHdr/@PACKAGETYPE	An attribute added by the Common Specification for describing the type of the IP. The vocabulary to be used contains values: • SIP • AIP • DIP • AIU • AIC The vocabulary is managed by the DAS Board and will be updated when required.	1..1
Agent	metsHdr/agent	The *metsHdr* must include at least one agent describing the software which has been used to create the package (TYPE=”OTHER” ROLE=”CREATOR” OTHERTYPE=”SOFTWARE”). Description of all other agents is optional.	1..n
Agent ID	metsHdr/agent/@ID	An ID for the agent.	0..1
Agent role	metsHdr/agent/@ROLE	The role of the agent. The Common Specification requires describing at least one agent with the *agent/@ROLE* value “CREATOR”. For other (optional) occurrences of *agent* this attribute shall use a value from the fixed list provided by METS.[21]	1..1
Other agent role	metsHdr/agent/@OTHERROLE	A textual description of the role of the agent in case the value of *agent/@ROLE* is “OTHER”.	0..1
Agent type	metsHdr/agent/@TYPE	The Common Specification requires that at least one instance of the *agent* element includes the *agent/@TYPE* attribute with the	0..1

[21] Values are: Creator, Editor, Archivist, Preservation, Disseminator, Custodian, IPowner, Other

		value “OTHER”. In other occurrences of the *agent* element the attribute is optional. If used, values defined in official METS documentation shall be followed (“individual”, “organisation”, “other").	
Other agent type	metsHdr/agent/@OTHERTYPE	The Common Specification requires that at least one instance of the *agent* element includes the *agent/@OTHERTYPE* attribute with the value “SOFTWARE”. In other occurrences this attribute shall only be used in case the value of agent/@TYPE is “OTHER”.	0..1
Agent name	metsHdr/agent/name	The name of the agent. In the Common Specification occurrence of the *agent* element this element must provide the name of the software tool which was used to create the IP.	1..1
Note about agent	metsHdr/agent/note	Additional information about the agent. We recommend using this element to provide version information for the tool which was used to create the IP.	0..1
Alternative ID for content	metsHdr/altRecordID	A container for an alternative ID for the package content.	0..n
ID of alternative record ID	metsHdr/altRecordID/@ID	An ID for the *altRecordID* element within the METS document.	0..1
ID type	metsHdr/altRecordID/@TYPE	Used to describe the type of ID assigned. It is recommended to use the Library of Congress vocabulary for this element when used.	0..1
METS document ID	metsHdr/metsDocumentID	A unique identifier for the METS document itself.	0..1
Document ID	metsHdr/metsDocumentID/@ID	The ID of the *metsDocumentID* element.	0..1
ID type	metsHdr/metsDocumentID/@TYPE	The type of the identifier assigned to the element.	0..1

Full example of the METS header:

```
<metsHdr CREATEDATE="2015-11-18T15:50:14" LASTMODDATE=“2015-11-28T13:24:56“>
  <agent TYPE="OTHER" ROLE="CREATOR" OTHERTYPE="SOFTWARE">
    <name>E-ARK SIP Creator</name>
    <note>VERSION=0.0.1</note>
  </agent>
</metsHdr>
```

5.3.3. Use of the METS descriptive metadata section (element *dmdSec*)

The purpose of the METS descriptive data section is to embed or refer to files containing descriptive metadata.

The Common Specification as such does not make any assumptions on the use of specific descriptive metadata schemas. As such, implementers are welcome to use descriptive metadata following any standards inside a Common Specification package.

Specific elements for which the exact use is fixed within this specification are highlighted in the following table.

Name	*Element/ Attribute*	*Description and usage*	*Cardinality*
Descriptive metadata section	dmdSec	Must be used if descriptive metadata about the package content is available. NOTE: According to official METS documentation each metadata section must describe one and only one set of metadata. As such, if implementers want to include multiple occurrences of descriptive metadata into the package this must be done by repeating the whole *dmdSec* element for each individual metadata.	0..n
dmdSec ID	dmdSec/@ID	Mandatory, identifier must be unique within the package	1..1
ID of metadata group	dmdSec/@GROUPID	Can be used to group together different metadata sections.	0..1
Reference to administrative metadata	dmdSec/@ADMID	In case administrative (provenance) metadata is available and described within METS about changes to the descriptive metadata, this element must reference the appropriate ID of the administrative metadata section.	0..1
Date created	dmdSec/@CREATED	Required by this specification. Creation date of the metadata in this section, needed to track changes to metadata files.	1..1
Metadata status	dmdSec/@STATUS	Status of the metadata. Recommended for use to indicate currency of package. If used it is recommended to use one of the two values “SUPERSEDED” or “CURRENT”.	0..1
External metadata link	dmdSec/mdRef	Reference to the descriptive metadata file stored in the “**metadata**” folder of the IP. In each occurrence of the *dmdSec* exactly one of the elements mdRef or mdWrap must be present. The Common Specification recommends the use of *mdRef* over *mdWrap*	0..1

Section ID	mdRef/@ID	Unique ID for the *mdRef* section within the METS document.	0..1
File mime type	mdRef/@MIMETYPE	The IANA media type for the external file.	0..1
File name	mdRef/@LABEL	A name for the referenced file.	0..1
File pointer	mdRef/@XPTR	Locates the point within a file to which the *mdRef* element refers, if applicable, using any valid XPointer scheme.	0..1
Locator type	mdRef/@LOCTYPE	Specifies the locator type used in the xlink:href which points to the file.	1..1
Other locator type	mdRef/@OTHERLOCTYPE	Required when *mdRef/@LOCTYPE* ="OTHER".	0..1
Type of metadata	mdRef/@MDTYPE	Specifies the type of metadata in the linked file. Values should be taken from the METS list provided.[22]	1..1
Type version	mdRef/@MDTYPEVERSION	The version of the metadata type described in MDTYPE	0..1
Other metadata type	mdRef/@OTHERMDTYPE	The type of metadata when MDTYPE="OTHER"	0..1
File size	mdRef/@SIZE	Size of linked file in bytes	0..1
File creation date	mdRef/@CREATED	Date the linked file was created	0..1
File checksum	mdRef/@CHECKSUM	The checksum of the linked file	0..1
File checksum type	mdRef/@CHECKSUMTYPE	The type of checksum used for calculating the checksum of the linked file	0..1
Link to embedded metadata files	mdWrap	Wrapper for descriptive metadata embedded into the METS document. In each occurrence of the *dmdSec* exactly one of the elements mdRef or mdWrap must be present. The Common Specification recommends the use of *mdRef* over *mdWrap*	0..1
Section ID	mdWrap/@ID	Unique ID for the mdWrap section within the METS document.	0..1
File mime type	mdWrap/@MIMETYPE	The IANA mime type for the wrapped metadata.	0..1
File name	mdWrap/@LABEL	A name for the associated metadata.	0..1
Type of metadata	mdWrap/@MDTYPE	Specifies the type of embedded metadata. Values should be taken from the METS list provided.[23]	1..1
Type version	mdWrap/@MDTYPEVERSION	The version of the metadata type described in MDTYPE	0..1
Other metadata	mdWrap/@OTHERMD	The type of metadata when	0..1

[22] Values available are: MARC, MODS, EAD, DC, NISOIMG, LC-AV, VRA, TEIHDR, DDI, FGDC, LOM, PREMIS, PREMIS:OBJECT, PREMIS:AGENT, PREMIS:RIGHTS, PREMIS:EVENT, TEXTMD, METSRIGHTS, OTHER. But note that the Common Specification (and METS) expects PREMIS metadata to be in the *amdSec* not *dmdSec*.

[23] Values available are: MARC, MODS, EAD, DC, NISOIMG, LC-AV, VRA, TEIHDR, DDI, FGDC, LOM, PREMIS, PREMIS:OBJECT, PREMIS:AGENT, PREMIS:RIGHTS, PREMIS:EVENT, TEXTMD, METSRIGHTS, OTHER. But note that the Common Specification (and METS) expects PREMIS metadata to be in the *amdSec* not *dmdSec*.

type	TYPE	MDTYPE=”OTHER”	
File size	mdWrap/@SIZE	Size of associated metadata in bytes	0..1
File creation date	mdWrap/@CREATED	Date the embedded metadata was created	0..1
File checksum	mdWrap/@CHECKSUM	The checksum of the wrapped content	0..1
File checksum type	mdWrap/@CHECKSUMTYPE	The type of checksum used for calculating the checksum of the embedded metadata	0..1
Binary data wrapper	mdWrap/binData	A wrapper element to contain Base64 encoded metadata	0..1
XML data wrapper	mdWrap/xmldata	A wrapper element to contain XML encoded metadata	0..1

Example of the METS <dmdSec> element using <mdRef>:

```
<dmdSec ID="ID74f5dd4e-0a83-49d7-af50-21a4cc974744">
   <mdRef
      MIMETYPE="application/xml"
      ID="IDa9abe6db-84eb-4af3-9d45-ca235a959312"
      MDTYPE="EAD"
      LOCTYPE="URL"
      xlink:href="file://metadata/descriptive/EAD.xml"
      xlink:type="simple"
      CREATED="2015-11-25T14:22:52"
      CHECKSUM="58b7855c94bb817af06bc969f7791b357c5ee22946981b8c18cc216384c25628"
      CHECKSUMTYPE="SHA-256" />
</dmdSec>
```

Kommenteeri: [KA1]: KÕIK ID näited korda teha!!!

5.3.4. Use of the METS administrative metadata section (element *amdSec*)

The purpose of the METS administrative data section is to embed or refer to files containing administrative metadata about the IP content objects. The Common Specification (and METS) categorises preservation metadata as administrative metadata, specifically Digital Provenance metadata, hence all preservation metadata should be referenced from a *digiprovMD* element within the *amdSec*.

The Common Specification allows both the embedding of metadata within the METS.xml file and keeping metadata in external files within the IP. Where preservation metadata is stored in external files (external to the METS file) it should be referenced using the *mdRef* element. Embedded metadata is wrapped using the *mdWrap* element. Note that for scalability concerns the Common Specification recommends the use of *mdRef* over *mdWrap*.

The METS *amdSec* element must include references to all relevant metadata either embedded or in external files located in the folder “**metadata/preservation**”. This means also that the root level METS.xml file must refer only to the root level preservation metadata and the representation METS.xml file must refer only to the representation level preservation metadata.

The specific requirements for the *amdSec* element, its sub-elements and attributes are presented in the following table.

Name	*Element/ Attribute*	*Description and usage*	*Cardin*

			ality
Administrative metadata	amdSec	In case administrative / preservation metadata is available, it must be described using the *amdSec* element.	0..n
Admin metadata ID	amdSec/@ID	Unique ID for the *amdSec* within the METS document	0..1
Provenance metadata	amdSec/digiprovMD	The Common Specification recommends the use of PREMIS metadata for recording information about preservation events. If used, PREMIS metadata must appear in a *digiprovMD* element, either embedded or linked. It is mandatory to include one *digiprovMD* element for each external file in the "**metadata/preservation**" folder, or for each embedded set of PREMIS metadata.	0..n
Technical metadata	amdSec/techMD	The use of *techMD* is not recommended. Instead, detailed technical metadata should be included into or referenced from appropriate PREMIS files	0..n
Rights metadata	amdSec/rightsMD	Optional. The Common Specification recommends including a simple rights statement which describes the overall access status of the package with the following values: • *Open* • *Closed* • *Partially closed* • *Not known*. However, the exact schema and element is up to individual implementations to decide	0..n
Source metadata	amdSec/sourceMD	Optional, no further requirements	0..n
The following attributes are available for use with each of the four specific metadata areas listed above (*xxx* below stands for *amdSec/digiprovMD*, *amdSec/techMD*, *amdSec/rightsMD* and *amdSec/sourceMD*.			
Metadata section ID	xxx/@ID	Mandatory for each of the four elements *amdSec/digiprovMD*, *amdSec/techMD*, *amdSec/rightsMD* and *amdSec/sourceMD*. Identifier must be unique within the package	1..1
Metadata group ID	xxx/@GROUPID	Optional, no further requirements	0..1
Reference to administrative metadata	xxx/@ADMID	In case administrative (provenance) metadata is available and described within METS about changes to the metadata occurrence described here, this element must reference the appropriate ID of the administrative metadata section.	0..1
Metadata creation date	xxx/@CREATED	Optional, no further requirements	0..1
Metadata status	xxx/@STATUS	Recommended for describing currency of metadata. If used, must include one of the	0..1

		two values “superseded” or “current”	
Metadata referenced in the *amdSec* should be linked using either *mdRef* when the metadata is in external files, or *mdWrap* when the metadata is embedded within the METS file. Use of *mdRef* and *mdWrap* is described under the *dmdSec* above and will not be repeated here.			

Full example of the METS <amdSec> element:

```
<amdSec ID="ID1a57e479-20e2-4e99-868b-88d0f816d109">
    <digiprovMD ID="ID41d8bb3c-f7c1-4254-aa9f-825009314fb0">
      <mdRef MIMETYPE="text/xml" xlink:href="file:metadata/preservation/premis1.xml"
LOCTYPE="URL" CREATED="2015-11-18T15:50:14"
CHECKSUM="8aa278038dbad54bbf142e7d72b493e2598a94946ea1304dc82a79c6b4bac3d5" xlink:type="simple"
ID="ID58ecdae0-b6af-4ad9-abf1-f6c2971f253a" MDTYPE="OTHER" CHECKSUMTYPE="SHA-256"/>
    </digiprovMD>
    <digiprovMD ID="ID7f7c41b9-e083-40b4-adf3-261d68e5e15b">
      <mdRef MIMETYPE="text/xml" xlink:href="file:metadata/preservation/premis2.xml"
LOCTYPE="URL" CREATED="2015-11-18T15:50:14"
CHECKSUM="70988d963a8f814be17ab1644bb5d3cc5f3ebb0b06d1e53482b90bf12f09b8e9" xlink:type="simple"
ID="IDf14692b6-d8f9-46e2-8e6d-5a409bd734f1" MDTYPE="OTHER" CHECKSUMTYPE="SHA-256"/>
    </digiprovMD>
  </amdSec>
```

5.3.5. Use of the METS file section (element *fileSec*)

Use of the METS *fileSec* element is highly recommended by the Common Specification (although not mandatory). It should describe all components of the IP which have not been already included in the *amdSec* and *dmdSec* elements. For all files the location and checksum need to be available. Therefore the main purpose of the METS file section is to serve as a “table of contents” or “manifest” and allow validating the integrity of the files included into the package.

The main requirement of the Common Specification is that the file section of both the root and representation METS files includes at least one file group (element *fileGrp*). This so-called “Common Specification file group” should follow the requirements below:

- The file group should be defined by a single *fileGrp* element
    - It is mandatory to use the @USE attribute with a fixed value of either “Common Specification root” (for the root METS file) or “Common Specification representation [representation ID]” (for the representation METS file if available)
    - *Example: <fileGrp USE=“Common Specification root”>*
- Each of the structural components (i.e. documentation, schemas, data) should be described by its own nested *fileGrp* element
    - The value of the @USE attribute of the nested *fileGrp* element should reflect the name of the folder (i.e. USE=“documentation”; USE=“data”; USE=“schemas”);
- In case representations include their own METS files, the components (including data files) of a representation should be described only in the representation METS. The root METS file should still include a *fileGrp* for each representation but only reference the METS.xml file of the representation.

The specific requirements for elements, sub-elements and attributes are listed in the following table. Note that use of the *stream* and *transformFile* elements are not discussed below. Implementers wishing to use either of these METS elements should follow the requirements in the METS documentation.

Name	***Element/ Attribute***	***Description and usage***	***Cardinality***
File section	fileSec	Recommended to include one *fileSec* element in each METS file	0..1
File section ID	fileSec/@ID	Recommended. The identifier must be unique within the METS file.	0..1
File group	fileSec/fileGrp	This specification requires that one specific occurrence of the *fileGrp* element is included as described above. Implementers are welcome to define and add additional file groups necessary for internal purposes. The main *fileGrp* element includes additional nested *fileGrp* elements, one for each folder of the package (except metadata described in *amdSec* and *dmdSec*).	1..n
File group ID	fileSec/fileGrp/@ID	Recommended, identifier must be unique within the package	0..1
File group version date	fileSec/fileGrp/@VERSDATE	Version date of the file grouping	0..1
Reference to administrative metadata	fileSec/fileGrp/@ADMID	In case administrative metadata is available and described within METS about the file group, this element must reference the appropriate ID of the administrative metadata section.	0..1
File group intended use	fileSec/fileGrp/@USE	Recommended in Common Specification with one occurrence bearing the values “Common Specification root” (for the root *fileGrp* element and the names of appropriate folders for nested *fileGrp* occurrences.	1..1
Files	fileSec/fileGrp/file	The Common Specification requires that *fileGrp* must contain at least one *file* element either pointing to content files with *FLocat* or wrapping the content files using *FContent*	1..n
File element ID	fileSec/fileGrp/file/@ID	Mandatory, must be unique across the package	1..1
Mime type of referenced file	fileSec/fileGrp/file/@MIMETYPE	The IANA mime type for the wrapped or linked file. Required by the Common Specification.	1..1
Fie sequencing	fileSec/fileGrp/file/@SEQ	Used to describe the sequence of files listed within the *fileGrp* element	0..1
File size	fileSec/fileGrp/file/@SIZE	Size of the linked or embedded file in bytes. Required by the Common Specification	1..1
Date file created	fileSec/fileGrp/file/@C	Date the embedded/linked file was created.	1..1

	REATED	Required by the Common Specification	
File checksum	fileSec/fileGrp/file/@CHECKSUM	The checksum of the embedded/linked file. Required by the Common Specification	1..1
File checksum type	fileSec/fileGrp/file/@CHECKSUMTYPE	The type of checksum used for the embedded/linked file. Required by the Common Specification	1..1
File owner	fileSec/fileGrp/file/@OWNERID	Unique ID of the file assigned by its owner	0..1
Reference to administrative metadata	fileSec/fileGrp/file/@ADMID	In case administrative metadata is available and described within METS about the file, this element must reference the appropriate ID of the administrative metadata section.	0..1
Related dmdSec ID	fileSec/fileGrp/file/@DMDID	Value for the ID attribute of the *dmdSec* containing metadata describing the content files listed in the file element.	0..1
Related group ID	fileSec/fileGrp/file/@GROUPID	Provides an ID for a *fileGrp* containing related files.	0..1
File intended use	fileSec/fileGrp/file/@USE	Statement about intended use of the files	0..1
File location	fileSec/fileGrp/file/FLocat	The location of each external file must be defined by the <FLocat> element using the same rules as for referencing metadata files. All references to files should be made using the XLink href attribute and the file protocol using the relative location of the file. Example: `xlink:href="file:schemas/mets.xsd"`	
File location ID	fileSec/fileGrp/file/FLocat/@ID	An ID for the <FLocat> element	0..1
File locator	fileSec/fileGrp/file/FLocat/@LOCTYPE	Mandatory locator pointing to the external file.	1..1
File locator type	fileSec/fileGrp/file/FLocat/@OTHERLOCTYPE	Description of the type of locator used	0..1
File intended use	fileSec/fileGrp/file/FLocat/@USE	Statement about intended use of the linked file	0..1
File content	fileSec/fileGrp/file/FContent	Used for identifying content files wrapped within the METS file. The content file must be either encoded in base64 and inside an <binData> wrapper, or encoded in XML and included within an <xmlData> wrapper.	0..1
Content file element ID	fileSec/fileGrp/file/FContent/@ID	An ID for the <FContent> element	0..1
File intended use	fileSec/fileGrp/file/FContent/@USE	Statement about intended use of the embedded file	0..1

Example of the *fileSec* element (root METS file):

```
<fileSec>
   <fileGrp USE="Common Specification root" ID="IDae911aa8-24f0-4bd8-a684-32044b89d687">
      <fileGrp USE="schemas" ID="IDae911aa8-24f0-4bd8-a684-32056b89d789">
         <file MIMETYPE="application/xsd" USE="Schema" CHECKSUMTYPE="SHA-256" CREATED="2015-12-
04T09:59:45" CHECKSUM="41d38f0a204e7dbda2838d93ad8eb5cf6bed92acd9c2f06f497faf47722e990d"
ID="ID04918b96-cf9f-41fa-ab13-3d550aaf94f5" SIZE="6814">
            <FLocat xlink:href="file://schemas/METS.xsd" xlink:type="simple" LOCTYPE="URL"/>
         </file>
      </fileGrp>
      <fileGrp USE="representations" ID="IDae055ba8-24f0-4bd8-a684-32056b89d882">
         <fileGrp USE="representation123" ID="IDbc911aa8-24f0-4bd8-a684-32056b89d789">
            <file MIMETYPE="application/xml" USE="Representation METS" CHECKSUMTYPE="SHA-256"
CREATED="2015-12-04T09:59:45"
CHECKSUM="41d38f0a204e7dbda2838d93ad8eb5cf6bed92acd9c2f06f497faf47722e990d" ID="ID04918b96-cf9f-
41fa-ab13-3d550aaf94f5" SIZE="6814">
               <FLocat xlink:href="file://representations/representation123/METS.xsd"
xlink:type="simple" LOCTYPE="URL"/>
            </file>
         </fileGrp>
      </fileGrp>
      <fileGrp USE="documentation" ID="ID7d136e4c-26fe-40da-85a2-67a42efd6b27">
      ...
      </fileGrp>
   </fileGrp>
</fileSec>
```

Example of the *fileSec* element (representation METS file):

```
<fileSec>
   <fileGrp USE="Common Specification representation representation123" ID="IDae911aa8-24f0-4bd8-
a684-32044b89d687">
      <fileGrp USE="data" ID="IDae911aa8-24f0-4bd8-a684-321556389d687">
         <fileGrp USE="user-defined-data-subfolder" ID="IDae911aa8-24f0-4bd8-a684-32044b89d789">
            <file MIMETYPE="application/pdf" USE="data" CHECKSUMTYPE="SHA-256" CREATED="2015-12-
04T09:59:45" CHECKSUM="41d38f0a204e7dbda2838d93ad8eb5cf6bed92acd9c2f06f497faf47722e990d"
ID="ID04918b96-cf9f-41fa-ab13-3d550aaf94f5" SIZE="6814">
               <FLocat xlink:href="file://data/user-defined-data-subfolder/contentfile.pdf"
xlink:type="simple" LOCTYPE="URL"/>
            </file>
         </fileGrp>
         ...
      </fileGrp>
      <fileGrp USE="documentation" ID="ID7d136e4c-26fe-40da-85a2-67a42efd6b27">
      ...
      </fileGrp>
   </fileGrp>
</fileSec>
```

5.3.6. Use of the METS structural map (element *structMap*)

The purpose of the METS structural map section is to provide an overview of ALL components of a Common Specification Information Package. It also links the elements of that structure to associated content files and metadata. It is a mandatory and ultimate means to define the full structure of the package – including metadata, representations, schemas, documentation and user added components and folders. In other words, tools compatible with the Common Specification will count on the information available within the *structMap* element as the primary means of identifying all components of the package. As such it is the

most crucial component for the validation of any Common Specification Information Package and must always be present.

The Common Specification Information Package requires the inclusion of one structural map according to the principles described below. However, implementers are welcome to define additional structural maps for their internal purposes by repeating the *structMap* element.

The most crucial requirements for the Common Specification mandated structural map are as follows:

- The *structMap* element has a mandatory attribute @LABEL which has the fixed value of "Common Specification structural map". The @LABEL attribute is used to distinguish the Common Specification mandated structural map occurrence from any other, user-defined, structural maps. As such we can also derive the requirement, that any user-defined structural maps must not use the LABEL value of "Common Specification structural map";
- The internal structure of the structural map (expressed by hierarchical *div* elements) follows the Common Specification physical structure as described in chapter 4, therefore grouping together metadata, representations, schemas, documentation and user-defined folders;
    - All *div* elements must use the attribute LABEL with the value being the name of the folder (as an example "**metadata**")
- In case both root and representation METS files exist, the structural map in the root METS file
    - Lists all files in all folders with the exception of the content of the representation folders
    - Lists all representations (as separate *div* elements)
    - Lists only the appropriate representation METS file using the *mptr* element as the content of the representation
- The structural map in a representation METS file lists all files within the representation with no exceptions

The specific requirements for elements, sub-elements and attributes are listed in the following table. Note that the *area*, *seq* and *par* elements are not discussed below.

Name	***Element/ Attribute***	***Description and usage***	***Cardinality***
Structural map	structMap	Each METS file needs to include exactly one *structMap* element used exactly as described in this table. Institutions can add their own additional custom structural maps as separate *structMap* sections.	1..n
Structural map ID	structMap/@ID	Optional, but if used must be unique within the package	0..1
Type of structural map	structMap/@TYPE	Mandatory in this specification. The value must be "physical"	1..1
Structural map name	structMap/@LABEL	Mandatory in this specification. The value must be "Common Specification structural map"	1..1
Structural divisions	structMap/div	Each folder (and sub-folder) within the package must be represented by an occurrence of the element. Please	0..n

		note that sub-folders must be represented as nested div elements.
		
		Example:
		
		`<structMap TYPE="physical" LABEL="Common Specification structural map">`
		`<div LABEL="Package123">`
		`<div LABEL="metadata">`	

Structural division ID	structMap/div/@ID	Mandatory, identifier must be unique within the package	1..1
Structural division type	structMap/div/@TYPE	No specific requirements	0..1
Structural division name	structMap/div/@LABEL	Mandatory, value must be the name of the folder ("**metadata**", "**descriptive**", "**schemas**", "**representations**", etc). The LABEL value of the first *div* element in the package is the ID of the package	1..1
Reference to descriptive metadata	structMap/div/@DMDID	ID attribute values identifying the *dmdSec*, elements in the METS document that contain or link to descriptive metadata pertaining to the structural division represented by the current *div* element	0..1
Reference to administrative metadata	structMap/div/@ADMID	No specific requirements	0..1
Structural division order	structMap/div/@ORDER	Not used in the specific Common Specification *structMap* occurrence	0
Structural division order name	structMap/div/@ORDERLABEL	Not used in the specific Common Specification *structMap* occurrence	0
Structural division content IDs	structMap/div/@CONTENTIDS	IDs for the content in this division. No specific use requirements.	0..1
File pointer	structMap/div/fptr	If the folder which is described by the *div* element includes computer files these must be referenced by using the *fptr* element. The only exception is the description of representations (see below for the use of *mptr*). The *fptr* child elements *par*, *seq* and *area* must not be used.	0..n
File pointer ID	structMap/div/fptr/@ID	No specific requirements	0..1
ID of content	structMap/div/fptr/@FILEID	Mandatory, must be the ID used in the appropriate *file* or *mdRef* element	1..1
File content IDs	structMap/div/fptr/@CONTENTIDS	IDs for the content referenced by this *fptr* element. No specific requirements	0..1
METS pointer	structMap/div/div/mptr	In the case of describing representations within the package (i.e.	0..n

		representations/representation1) the content of the representations must not be described. Instead the of the specific representation should include one and only one occurrence of the <mptr> element, pointing to the appropriate representation METS file. The references to representation METS files must be made using the XLink href attribute and the file protocol using the relative location of the file. Example: `xlink:href="file:representation/representation1/mets.xml"` The XLink type attribute is used with the fixed value “simple”. Example: `xlink:type="simple"` The LOCTYPE attribute is used with the fixed value ”URL”	
METS pointer ID	structMap/div/div/mptr/@ID	Unique ID for this element	0..1
METS pointer	structMap/div/div/mptr/@LOCTYPE	The locator type used in the xlink:href attribute	0..1
METS xlink type	structMap/div/div/mptr/@OTHERLOCTYPE	Locator type in xlink:href when LOCTYPE=”OTHER”	0..1
METS pointer content IDs	structMap/div/div/mptr/@CONTENTIDS	The content ID for the content represented by the *mptr* element.	0..1

Full example of the Common Specification *structMap* element (root METS file):

```
<structMap TYPE="physical" LABEL="Common Specification structural map">
   <div LABEL="9da99df7-2237-48d6-90ef-01d99447c16f">
      <div LABEL="metadata">
         <div LABEL="descriptive">
            <fptr FILEID="IDc04f8f55-802e-4646-b5f9-78b8e864e530"/>
            <fptr FILEID="IDa2da0aa8-bf9c-4a79-a83d-2944cb2031ab"/>
         </div>
         <div LABEL="preservation">
            <fptr FILEID="IDc2ccef19-802e-4646-b5f9-78b8e864e532"/>
            <fptr FILEID="IDa2da11a8-bf9c-4a79-a83d-2944cbfee654"/>
         </div>
      </div>
      <div LABEL="schemas">
         <fptr FILEID="ID845a7a5b-0cfe-43ff-acd9-14f5f0463e28"/>
      </div>
      <div LABEL="representations"/>
         <div LABEL="representations/aip-docs_mig-1">
            <mptr xlink:href="file://representations/aip-docs_mig-1/METS.xml" xlink:type="simple"
LOCTYPE="URL"/>
```

```
            </div>
            <div LABEL="representations/aip-imgs_mig-1">
               <mptr xlink:href="file://representations/aip-imgs_mig-1/METS.xml" xlink:type="simple"
LOCTYPE="URL""/>
            </div>
         </div>
      </div>
</structMap>
```

5.3.7. Use of the METS Structural Link Section (element *structLink*) *and Behavior Section* (element *behaviorSec*)

The Common Specification Information Package poses no additional requirements on the METS *structLink* and *behaviorSec* elements.

5.4. Use of PREMIS in a Common Specification Information Package

The Common Specification recommends and advocates the use of the PREMIS metadata standard for recording preservation and technical metadata about digital objects contained within Common Specification IPs. The Common Specification implements version 3.0 of the PREMIS Data Dictionary.[24] Note that use of PREMIS is not mandatory because a SIP will not always be able to include preservation metadata.

Although the Common Specification allows both the embedding of metadata within the METS file, and its inclusion in the IP in a separate metadata file, we strongly recommend keeping PREMIS metadata in discrete PREMIS XML files inside the IP. If PREMIS metadata is included in the IP in separate files, the naming and numbering of the PREMIS files are not restricted, meaning that implementations can choose to either store all preservation metadata in a single PREMIS file or split them into multiple files. The only requirement in this case is that all PREMIS files must be listed in the appropriate METS file, i.e. root PREMIS files from the root METS file and representation PREMIS files from the representation METS files, and referenced in the METS file(s) using the *mdRef* attributes and elements.

Therefore, the main recommendation of the Common Specification is that preservation metadata are included in the information package in PREMIS format. Although this is not mandatory, all tools claiming to be able to validate Common Specification IPs must also be able to validate PREMIS metadata once it exists within the package. The two high level requirements for use of PREMIS in Common Specification IPs are that:

- All preservation metadata is created according to official PREMIS guidelines[25];
- All PREMIS metadata is either embedded in or referenced from the *amdSec/digiprovMD* element of the appropriate METS file.

[24] PREMIS Data Dictionary for Preservation Metadata, version 3.0: http://www.loc.gov/standards/premis/v3/premis-3-0-final.pdf
[25] http://www.loc.gov/standards/premis/

Further, to enhance the interoperability scope of the Common Specification for Information Packages and to strengthen management of IPs in an archive, this specification imposes additional requirements in regard to use of PREMIS for describing Common Specification IPs. The principles adopted in the Common Specification for deciding the additional PREMIS semantic units required are:

- PREMIS should be used to record detailed technical metadata. In METS the technical metadata included should only include the checksums and size of files;
- As much technical information as possible should be included in PREMIS metadata by using extension schemas;
- Information about agents carrying out preservation actions should be recorded in PREMIS metadata and not in METS. The use of METS agents should be limited to those agents who are relevant for generic IP level events (for example, the creation of the package, submitting agency);
- Event descriptions should be included in PREMIS metadata as much as possible. Use of the official PREMIS event vocabulary is recommended[26];
- Detailed rights information should be included in PREMIS and not described in METS. The METS file should only include information about the whole package – is it totally open, partially restricted, needs review etc.[27] Where high level rights information in METS indicates restrictions, detailed, object-specific, rights information will be included in the PREMIS metadata;
- File format information for all files should be included as PUID[28] values in the appropriate PREMIS semantic units.

In addition to the mandatory semantic units required by PREMIS itself, the Common Specification requires, based on the requirements specified above, the following additional semantic units:

Name	Code	Semantic Unit Name	Rationale	Cardinality
File checksum	1.5.2	fixity	PREMIS requires the use of the *objectCharacteristics* semantic unit but leaves use of the fixity component optional. The Common Specification requires fixity for validation of the structure and content of IPs.	1..n
Relationship to other content	1.13	relationship	Required by the Common Specification for structural, provenance and contextual purposes.	1..n
Event outcome	2.5	eventOutcomeInformation	The only place to record the outcome of an event. Needed for authenticity. One of the sub elements, *eventOutcome* or *eventOutcomeDetail*, is required.	1..n
Linked agent ID	2.6	linkingAgentIdentifier	The Common Specification strongly recommends that most agent information be recorded in PREMIS metadata rather than METS, this semantic unit is required for authenticity and archival management purposes.	1..n
Linked object ID	2.7	linkingObjectIdentifier	Because event information is recorded in PREMIS, the Common Specification requires this semantic	1..n

[26] http://id.loc.gov/vocabulary/preservation/eventType.html

[27] Cf. Chapter "Use of the METS administrative metadata section (<amdSec>)"

[28] PUID stands here for "PRONOM Persistent Unique Identifier". See http://www.nationalarchives.gov.uk/PRONOM for more information.

			unit to link preservation events to objects for contextual metadata and audit logging purposes.	
Agent Name	3.2	agentName	Required by the Common Specification for recording human readable names for the agents associated with archival events performed on objects.	1..n

Vocabularies

This specification does not present a definitive list of vocabularies for use with PREMIS semantic units but does recommend the use of the Library of Congress vocabularies developed specifically to provide values for various PREMIS semantic units.[29]

In PREMIS each of the entities (objects, events, agents, rights) are identified by a generic set of identifier containers. These containers follow an identical syntax and structure consisting of an [entity]Identifier container holding two semantic units:

- [entity]IdentifierType
- [entity]IdentifierValue

The PREMIS data dictionary recognizes that the use of identifier types is an implementation specific issue and does not recommend or require particular vocabularies for identifier types. The Library of Congress has developed its own identifier type vocabulary[30] and the Common Specification recommends its use in lieu of implementation specific identifier type vocabularies, where these have not yet been developed.

[29] Available from http://id.loc.gov/vocabulary/preservation.html
[30] See http://id.loc.gov/vocabulary/identifiers.html

6. Implementation considerations

This chapter touches on some additional issues which are relevant in respect to implementing the Common Specification in real-life scenarios.

6.1. Content Information Type Specifications

6.1.1. What is a Content Information Type Specification?

The concept of Content Information Type Specification is essentially an extension method which allows for widening the interoperability scope of the Common Specification into a content specific level.

As defined by the OAIS Reference Model, *Content Information* is *"A set of information that is the original target of preservation or that includes part or all of that information. It is an Information Object composed of its Content Data Object and its Representation Information"*.

A *Content Information Type* can therefore be understood as a category of Content Information, for example relational databases, scientific data or digitised maps. And finally a *Content Information Type Specification* defines in technical terms how data and metadata (mainly in regard to the *Information Object*) must be formatted and placed within a Common Specification Information Package in order to achieve interoperability in exchanging specific *Content Information*.

As such, the following elements can be at the core of a Content Information Type Specification:

- The required file format of data;
- Description of how data must be placed and structured within the Common Specification folder structure (i.e. a sub-structure for the **"Data"** folder);
- Clearly defined requirements for specific representation metadata that needs to be available in PREMIS for rendering and understanding the *Content Data Object* appropriately;
- Clearly defined list of specific (binary) documentation or other components (like software, emulators, etc.) which have to be available for rendering and understanding the *Content Data Object* appropriately.

However, for practical purposes it is not sufficient to only deal with the *Information Object.* Especially for complex *Content Information Types* and large IPs it might also be relevant to describe explicitly requirements for other metadata (descriptive, administrative) which are relevant and crucial only for this specific content type. For example, the SMURF Content Information Type Specification, developed within the E-ARK project, does set specific requirements for how data (i.e. computer files) need to be referenced from descriptive metadata (in EAD format) in order to guarantee the integrity of data and metadata. Setting these requirements in a central specification will allow archival institutions to receive SIPs including ERMS extracts or whole systems and still be able to understand and validate the potentially complex structure of the whole data and metadata composition within it.

Concluding from the previous we can also see that Content Information Type Specification can potentially also be sector specific, and that there might be multiple specifications to cover a single content type. For example, archival institutions would be able to define a Content Information Type specification for

archiving web sites along with descriptive metadata in EAD format, while libraries might define a specification for archiving web sites along with metadata in MARC.

6.1.2. Maintaining Content Information Type Specifications

The number of possible Content Information Type Specifications is potentially unlimited. As well, it is the intention of the authors of this Common Specification to allow everybody in the wider community to create new specifications.

The maintenance of such a living environment is the role of the DLM Archival Standards Board (DAS Board, see www.dasboard.eu). The core principles of the maintenance regime are as follows:

- The DAS Board is responsible for establishing reasonable guidelines and quality requirements for new Content Information Type specifications, and publishing these on the Board website;
- The Board has the responsibility and mandate to manage a registry of available Content Information Type specifications which meet the guidelines and quality requirements;
- The Board does NOT take ownership of and have responsibility of maintaining and sustaining any Content Information Type specifications;
- There shall be no limitations to who is allowed to propose additional Content Information Type specifications;
- To ensure good quality of available specifications, the Board validates each proposed specification against the guidelines and quality requirements mentioned above. The validation shall be carried out free of charge and within a reasonable timeframe.

6.2. Handling large packages

By default a Common Specification IP is supposed to reside in a single folder or file (in case compression has been applied). However, the amount of data and metadata within a single IP can easily grow into sizes of several GB or even TB and as such can become difficult to manage and inefficient to process because, for example, of lacking media capacity.

The Common Specification itself can in principle be extended in multiple ways to support the segmenting of large packages into more manageable physical pieces. This chapter describes one way which exploits the Common Specification "representation METS" concept and extends it into a physical segmentation scenario.

However, it is worth noting that this is a "recommended approach" and is, at this point in time, not a part of the core Common Specification, as such it is also not expected that all tools support such a mechanism.

6.2.1. The structure for IP, their representations and their segments

According to the E-ARK Common Specification for IPs an IP can have several representations. All representations contain the same intellectual content, but as the name implies is another representation; in its most simple form this could be another file format such as TIFF instead of JPEG.

The segmenting approach described here is based on the following considerations:

- Most of the size of an IP is the content (data) which according to the Common Specification resides in the representations folder of the IP. As such also any segmenting should take place within the representations layer of the Common Specification;
- According to the Common Specification each representation is essentially a Common Specification IP itself, as it can consist of a METS metadata file, data, metadata, and any additional components;
- A segment of an IP must also be in the Common Specification format, i.e. it shall be possible to validate each individual segment as a Common Specification IP;
- Each IP shall consist of a parent segment (including at least the root METS file) and any number of child segments;
- It shall be possible to add new physical child segments (as an example a new representation) to the whole IP without having to update other child segments.

6.2.2. Using METS to refer from parent IP to child IP(s)

The method used to refer from parent to child is based on the ID of the IP of the child.

One reason for using ID and not URL or other more direct references to a location of the referenced METS file is the flexibility it gives to move the segmented IPs around in different storage locations. This is a flexibility often needed for segmented IPs that accumulated can be very large.

The value of the xlink:href attribute in the <mptr> element in the METS file of the parent IP is used.

This value is to be set to the value of the OBJID attribute of the <mets> element in the METS file of the child IP. According to the Common Specification, the OBJID attribute must have the value of the ID of the IP.

This is therefore sufficient for having the parent know the ID of the child, but the parent does not know the exact child location.

6.2.3. Using METS to refer from child IP to parent IP

The optional reference from child to the parent is based on the ID of the IP of the parent.

The value of the xlink:href attribute in <mptr> element in the METS file of the child IP is used.

This value is to be set to the value of the OBJID attribute of the <mets> element in the METS file of the parent IP. According to the Common Specification, the OBJID attribute must have the value of the ID of the IP.

This is therefore sufficient for having the child know the ID of the parent, but the child does not know the exact parent location.

6.2.4. An example for the Northwind database

Here follows a partial example, where the value of the xlink:href attribute in the <mptr> element (inside the element inside the <structMap> element) is "ID.AVID.RA.18005.rep0.seg0" after the urn NID part (urn:<NID>:<NSS>).

The value "ID.AVID.RA.18005.rep0.seg0" must now match the value of the OBJID attribute for the <mets> element in the child IP root METS file.

(Note that in order to save space in this example the CS mandatory ID attribute for the elements have been left out.)

Parent METS file

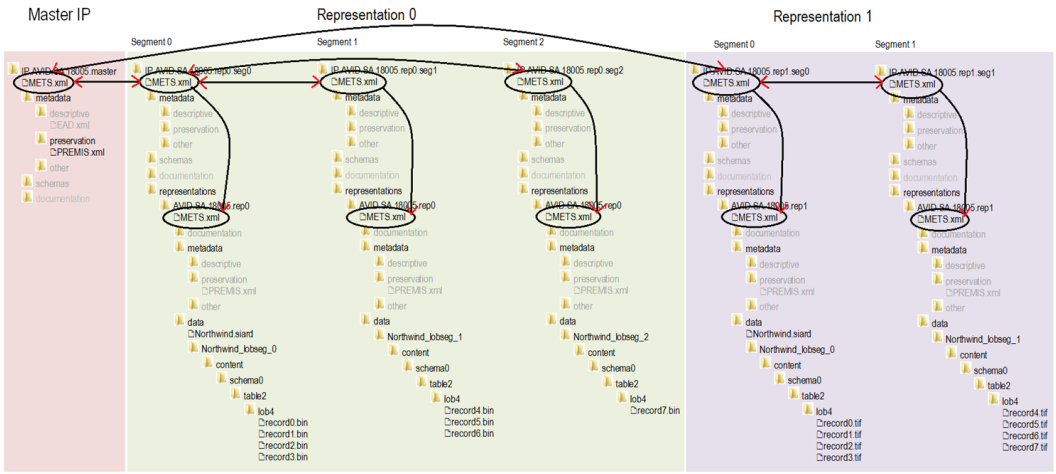
```
<div LABEL="representations">
  <div LABEL="representations/ID.AVID.RA.18005.rep0" ORDER="0" >
    <div LABEL="child IP" TYPE="representation child">
      <mptr xlink:href="urn:sa.dk:ID.AVID.RA.18005.rep0.seg0" xlink:title="root level METS file for representation 0" xlink:type="simple"
LOCTYPE="URN"/>
    </div>
  </div>
  <div LABEL="representations/ID.AVID.RA.18005.rep1" ORDER="1">
    <div LABEL="child IP" TYPE="representation child">
      <mptr xlink:href="urn:sa.dk:ID.AVID.RA.18005.rep1.seg0" xlink:title="root level METS file for representation 1" xlink:type="simple"
LOCTYPE="URN"/>
    </div>
  </div>
</div>

<!-- this top root level METS.xml IP only refers to the root level METS files in the representations using the <mptr> element -->
<!-- we use the attribute LABEL value 'child IP' in the 'div' element for representations in accordance with the AIP spec.3.3.1.9 -->
<!-- each root level METS file in the representations refer to its own METS files in the segments and in the representations folder using
the <mptr> element -->
<!-- we use the attribute LABEL value 'segment' in the 'div' element for representations-->
<!-- we have no CHECKSUM for these METS files because the mptr does not allow it and because the <file> element in the <fileGrp> in
the <fileSec> apparently is only to be used for files inside the package -->
<!-- the value of the attribute LABEL is the ID of the representation -->
<!-- representation 0 - images in jpg format -->
<!-- this is a METS reference to another METS file, and this file is in another segment - compare with CS v0.13 sec. 5.2, p 36 -->
<!-- the value of the attribute LABEL is the ID of the representation -->
<!-- representation 1 - a migration to tif -->
<!-- this is an indirect METS reference to another METS file, and this file is in another segment - compare with CS v0.13 sec. 5.2, p 36 --
>
```

Child METS file

```
<mets xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.loc.gov/METS/"
xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.loc.gov/METS/ schemas/mets.xsd"
PROFILE="http://www.ra.ee/METS/v01/IP.xml" TYPE="Database segment child" OBJID="ID.AVID.RA.18005.rep0.seg0" LABEL="root
level METS file for a representation segment">
..
..
..
  <div LABEL="parent IP" TYPE="Godfather IP"> <!-- working title - maybe master IP is more appropriate -->
    <mptr xlink:href="urn:sa.dk:ID.AVID.RA.18005.godfather" xlink:title="root level METS file for godfather IP" xlink:type="simple"
LOCTYPE="URN"/>
<!-- this is an indirect METS reference to another METS file. However, the referenced file is in another segment -->
  </div>
```

6.2.5. Illustration of references between METS files in a segmented IP

We need to segment an IP at the data folder in the representations level, but according to the Common Specification this can only be done at the IP level. Therefore this IP has been segmented at the top IP level, and not at the representations level.



Please note the following about the example:

- The Master IP MUST NOT contain representations
- A representation MAY be segmented
- The IDs are not just unique but haves implicit value for example purposes only
- In representation 0 the limits on folder size and amount of files requires three segments (0, 1 and 2)
- In representation 1 these limits have been increased and we only need two segments. Further the .bin files have been migrated to .tif.

6.3. Handling descriptive metadata within the Common Specification

Descriptive metadata are used to describe the intellectual contents of archival holdings, and they support finding and understanding individual information packages. The Common Specification allows essentially for the inclusion of any kind of descriptive metadata in the IP. However, it is required that all descriptive metadata must be placed into the **"metadata"** folder of the IP, and that it is recommended (should) to also exploit the possibility of creating a specific sub-folder **"descriptive"** as seen in Figure 11 below (cf. EAD.xml).

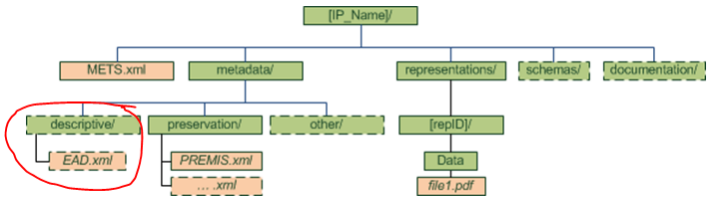


Figure 11: E-ARK IP descriptive metadata

Further, all descriptive metadata need to be described in and **referenced** from METS metadata (i.e. the METS.xml file) using the element <dmdSec> (Figure 12) and as such descriptive metadata are **not to be embedded** into the METS file directly.

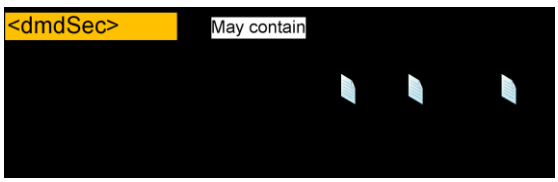


Figure 12: METS descriptive metadata

Following the requirement of explicitly and physically separating descriptive metadata and data we would also like to note, that for interoperability purposes appropriate descriptive metadata elements must explicitly refer to the data content they describe (unless the whole data portion is a single intellectual unit described as a discrete set of descriptive metadata). For example, the E-ARK project has explicitly defined that the EAD <dao> and <daogrp> elements shall be used to refer to content files from the descriptive metadata. However, regardless of the descriptive metadata standard in question the references from descriptive metadata must always follow the requirement posed in Chapter 5.2 above (i.e. create references according to the format defined in RFC 3986, or to express references as a relative path to the data files).

Finally we would also note that the recommendation of the Common Specification is to always include detailed metadata about intellectual access restrictions and copyright into descriptive metadata (i.e. not into the METS or PREMIS portions of the IP).

6.4. Technical requirements for Common Specification validation

Will be available by the 1st of February 2017.

Annex I: Full XML Examples

TBD

Until the end of the E-ARK project IP examples are maintained on a continuous basis on https://github.com/eark-project/information-package/tree/master/examples