INTERNAL DELIVERABLE

Project Acronym:	**E-ARK**
Grant Agreement Number:	**620998**
Project Title:	**European Archival Records and Knowledge Preservation**

DELIVERABLE DETAILS

DELIVERABLE REFERENCE NO.	**N/A**
DELIVERABLE TITLE	Introduction to the Common Specification for Information Packages in the E-ARK project
REVISION	**0.13**

AUTHOR(S)	
Name(s)	Organisation(s)
Tarvo Kärberg	National Archives of Estonia (NAE)
Karin Bredenberg	National Archives of Sweden / ES Solutions (ESS)
Björn Skog	ES Solutions (ESS)
Anders Bo Nielsen	Danish National Archives (DNA)
Kathrine Hougaard Edsen Johansen	Danish National Archives (DNA)
Alex Thirifays	Danish National Archives (DNA)
Sven Schlarb	Austrian Institute of Technology (AIT)
Andrew Wilson	University of Portsmouth / University of Brighton
Kuldar Aas	National Archives of Estonia (NAE)

Project co-funded by the European Commission within the ICT Policy Support Programme Dissemination Level		
P	**Public**	X
C	**Confidential, only for members of the Consortium and the Commission Services**	

REVISION HISTORY AND STATEMENT OF ORIGINALITY

Submitted Revisions 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 E-ARK implementation, include comments from Sven, Jan (AIT) and Andrew (UPHEC)
0.13	05.01.2015	Kuldar Aas, all	NAE, all	Update to include additional comments from E-ARK WPLs and Common Specification group members

Statement of originality:
This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENT

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.

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 E-ARK project;
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 that the (E-ARK) tools implementing the Common Specification can be taken up by institutions without needing further modifications or adaptations.

1.1. Common Specification and OAIS Information Packages

According to OAIS 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).

The Common Specification aims to summarise the common aspects of all these IPs. The main goal is to define the common IP aspects which are needed by tools for carrying out initial validation and identification of any package – for example the structure of the IP; location of data and metadata; core principles for preservation metadata. At the same time it is not in the scope of the Common Specification to standardise the elements of an IP which are not crucial for interoperability. Most crucially standardisation of resource discovery (i.e. descriptive) metadata is outside the scope of the Common Specification.

However, it is clear that due to the differences in needs regarding the submission, archival and dissemination processes it is not possible to have a Common Specification that covers all the aspects necessary for interoperable SIPs, AIPs and DIPs. The detailed (S/A/D)IP specifications share the Common Specification as the backbone but may vary in e.g. the specification of certain metadata elements.

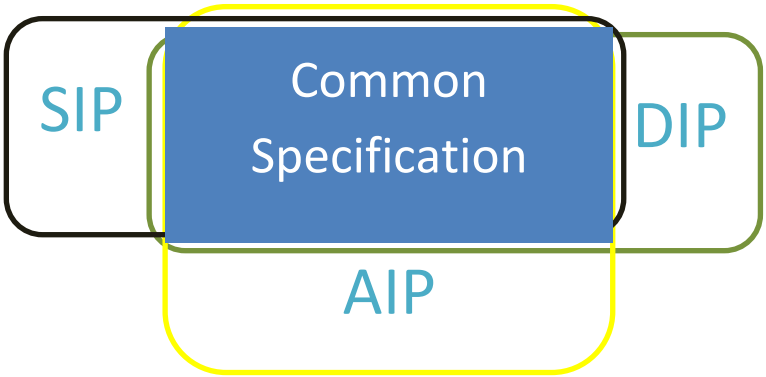


Figure 1: The scope of Common Specification in regard to OAIS Information Packages.

In addition to the Common Specification the E-ARK project therefore also defines separate SIP, AIP and DIP profiles[1] which extend the Common Specification with details specific for submission, storage and access scenarios.

Finally we would also like to note that as the main intention of the Common Specification is to support interoperability then the purpose of the AIP specification is different. Namely, within digital preservation systems the function of the AIP is less about supporting interoperability and more about the efficiency of active preservation processes in the specific storage solution. Therefore, implementers might prefer alternative requirements to those presented in this document. This is also the case within E-ARK as the AIP format deviates from the Common Specification in some aspects where interoperability and efficiency contradict.

However, we encourage preservation system providers to support the E-ARK Common Specification and AIP format as the means for repository migration to reduce the need for carrying out expensive export-ingest procedures, or to make it easier to use parallel preservation systems in order to reduce preservation risks.

1.2. Common Specification and Content Type Specifications

As an interoperability standard the Common Specification must be possible to be used 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 Type Specifications***. A Content Type Specification can include detailed requirements on how content, metadata and documentation for specific content types (as an example relational databases or geospatial data) has to be handled using the Common Specification.

The E-ARK project defines the following specific Content Type Specifications:

- Data sets: generic specification for encapsulating any type of data into the Common Specification;
- Relational databases (based on the SIARD 2[2] format);
- Electronic Records Management Systems (ERMS, based on the MoReq 2010 metadata model[3]);
- Geo-information (based on metadata guidance presented in the INSPIRE directive).

These four Content Type profiles will first become available in early 2016.

At the same time the E-ARK project is working on a management regime which would allow for the creation of additional Content Type Specifications after the end of the project and/or by external bodies. As an example, there is already some interest by ISO to create a Content Type Specification for e-publications around the EPUB 3.0 format. The management regime will be developed during 2016 in close collaboration between the project and all interested external bodies.

[1] The relevant E-ARK deliverables (D3.2 for SIP, D4.2 for AIP and D5.2 for DIP) are available at http://www.eark-project.com/resources/project-deliverables. New versions of the E-ARK SIP and AIP formats will be published in February 2016, while a new version of the DIP format will be published in April 2016.
[2] http://eark-project.com/resources/specificationdocs/32-specification-for-siard-format-v20
[3] http://www.moreq.info/index.php/specification

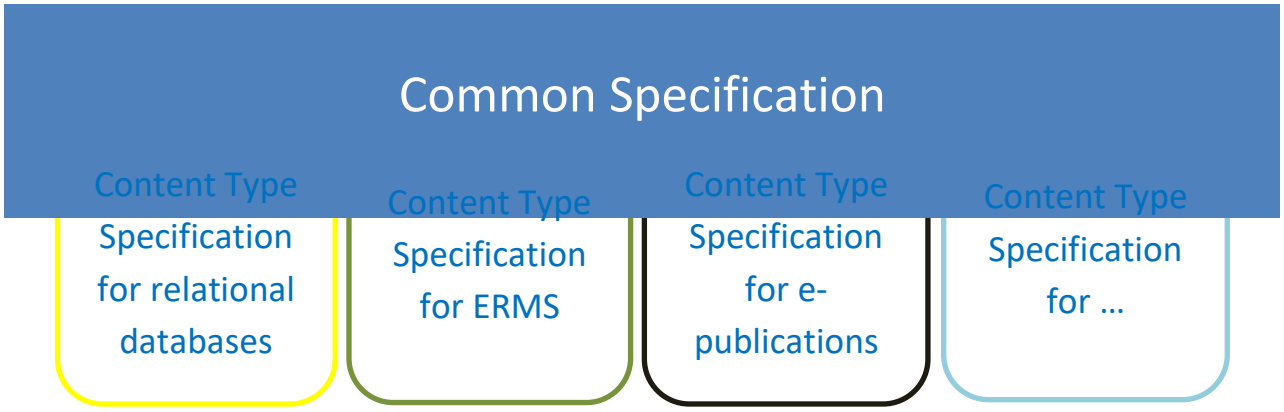


Figure 2: Common Specification and Content Type Specifications

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

To bring an overview of the specifications that the E-ARK project will provide, the following 3-layered illustration shows the Common Specification as the foundation, on top of which the SADIP extensions are built. Each of these contains one of the four types of content profiles that are specified in the E-ARK's Content Type Specifications.

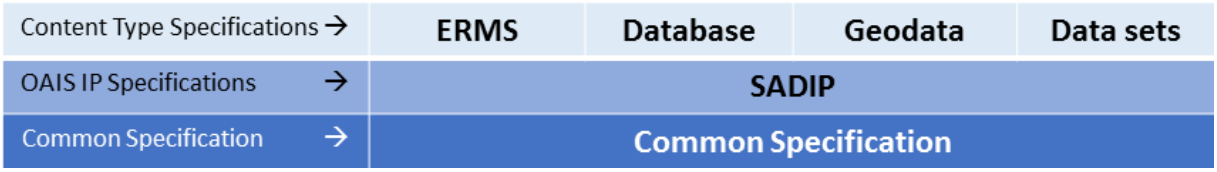


Figure 3: Overview of relations between the Common Specification; SIP, AIP and DIP specifications and Content Type Specifications

1.4. Relation to other documents

The 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*

    The E-ARK project 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 Methodology Abstract Standard (PAIMAS), 2004, public.ccsds.org/publications/archive/651x0m1.pdf*

    E-ARK has looked the four phases (Preliminary, Formal Definition, Transfer and Validation) of PAIMAS, their aims and expected results and decided that there are no obvious requirements for common information packages specification.

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

  E-ARK has 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 E-ARK reports (see below).

- *e-SENS (Electronic Simple European Networked Services) project, http://www.esens.eu/*

  E-ARK has investigated the e-Delivery and e-Documents related work in e-SENS and made sure that work within E-ARK is not duplicating theirs or producing any conflicts between deliverables.

- E-ARK deliverables[4]
  - 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 D3.2, E-ARK SIP Draft Specification
  - Deliverable D4.2, E-ARK AIP Draft Specification
  - Deliverable D5.2, E-ARK DIP Draft 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 introduces the Common Specification and a practical implementation of it. 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. The E-ARK project hopes that it is possible to extend this work beyond the project and, in long term, will be used as a conceptual model for any Information Package implementation across the globe. In more detail:

- **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 the previous chapter (Chapter 2). Based on these requirements a set of high-level solutions are proposed regarding for example the structure and use of metadata within any Information Package.

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

The second part of this document (Chapters 4 and 5) presents a practical implementation of the principles described in previous chapters. The main short-term aim of this part is to reach a sufficient level of technical and semantic interoperability within the E-ARK project to allow for the common development of practical tools. However, in long-term we also hope that the solutions described in these chapters are going to be taken up beyond the project to ultimately achieve global interoperability. In more detail:

- **Chapter 4** presents a detailed view of the structure of the E-ARK Information Packages, which underpins practical implementation.
- **Chapter 5** presents a detailed overview of metadata requirements within E-ARK 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.

Finally the current version of the document concludes with **Chapter 6 (Next steps on the Common Specification)** which describes future updates and additions already planned to this document within the remainder of the E-ARK project.

PART I: Common Specification for Information Packages

In this part of the document we build the argumentation 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.*

According to the international records management standard, ISO 15489, a record can be created and managed, whatever its technical format. Across all formats and systems it must be securely retained and managed for defined periods of time in accordance with an organizational policy for a variety of reasons, for example:

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

A digital production workflow, where the primary task is to keep information available at any given time both for the short and long-term, requires the use of long-term information management. As of now most implementations separate the short-term and long-term management of information into different systems – business and records systems on one hand and archival systems on the other.

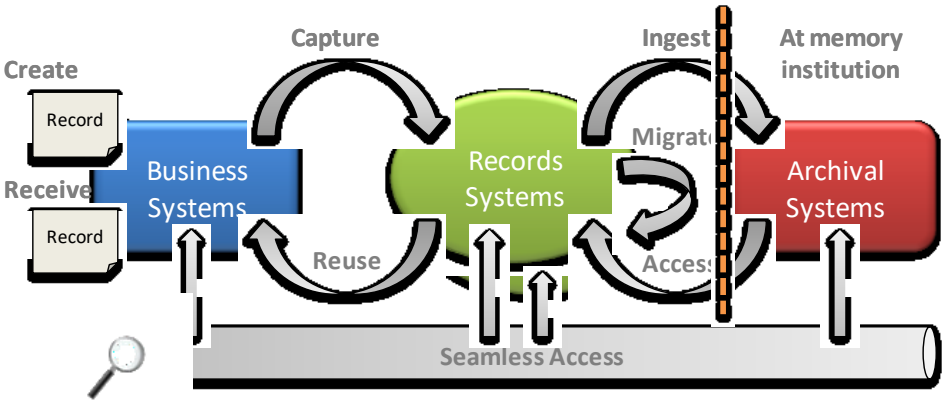


Figure 4: Information flow between live and archival systems

The implication for system owners 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
- without limiting the possibilities for discovering and reusing the information.

This document proposes a common specification for how data and metadata should be structured and packaged when transferred to archival systems, preserved in these and re-used. Such a specification, when implemented by a crucial number of archival and source system providers as the de facto standard, will give all users the assurance that no matter which archival system is in use, information can be received from and reused by a variety of different systems with no major effort.

The common specification can also serve as the central interoperability guideline allowing individual institutions to collaborate in creating specific tools for the transfer, preservation and access to archived content, therefore leading to decreased development costs for any single institution. As well, the common specification increases the possibility of easily changing products that are used and therefore gives suppliers an extended market to operate in.

3. Requirements for Information Packages

The core of any standardisation activity is to achieve a common understanding on what needs to be standardised and for what purpose. 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 use.

Another crucial factor to take into account is long-term sustainability. As such the requirements below are technology independent and designed in such a way that they could be implemented using any given technological components.

Ultimately the requirements below present a conceptual view of an Information Package, including an overall data model of it, use of data and metadata. An implementation of this conceptual view is presented in later chapters (4 - 5) of this document.

The requirements are mainly based on the results of the best-practice survey which was carried out in the initial stages of the project and is described in three separate deliverables (D3.1, D4.1 and D5.1). The needs identified in the survey have been further broadened and discussed within the project as well as with core external stakeholders.

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, Should, Could, Would) prioritisation statement[5]. Next to the short description the requirements also include a short rationale which describes the reason and background of the requirement.

Note for review: Please check the requirements and MoSCoW criteria carefully, comment if something is missing or unnecessary! Bear in mind that the main driver for the requirements must always be interoperability!

3.1. General requirements

Requirement 1.1: *The Common Specification MUST allow for the inclusion of any data or metadata, regardless of its type or format, in the Information Package.*

In order to achieve widespread use, and enable interoperability all implementations of the Common Specification need to support all kinds of data and metadata. If an Information Package definition fails to meet this requirement it is not possible to use it across different sectors and tools, therefore limiting practical interoperability.

Requirement 1.2: *The Common Specification MUST support the transfer of the Information Package by any means, methods or tools*

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

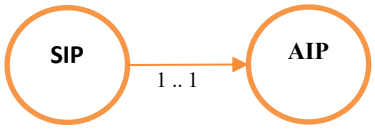
Tools and methods for transferring Information Packages between locations are constantly evolving. It is also possible that different methods might be preferred especially for packages of varying sizes.

To support this requirement the Common Specification does not define the use of a specific 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 (like the BagIt[6] envelope) on top of the Common Specification.

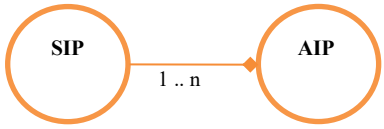
Requirement 1.3: *The Common Specification MUST not restrict the cardinality of transformations between SIPs, AIPs and DIPs.*

Different institutions have different solutions in regard to defining the transformations between SIPs and AIPs:

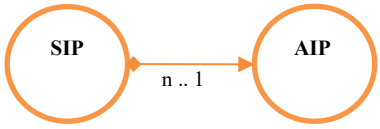
- One SIP can be transformed into exactly one AIP:



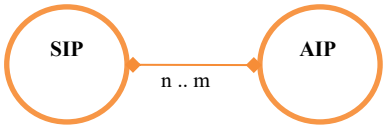
- Or one SIP is transformed into many AIPs:



- Or many SIPs are transformed exactly into one AIP:



- Or many SIPs are transformed into many AIPs:



Of course the same variations are possible for AIP to DIP transformations. The Common Specification does not define what should, for example, constitute a SIP or whether 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".

Further, the transformations of SIP(s) to AIP(s) and AIP(s) to DIP(s) are internal operations of digital archives and therefore not interoperability oriented. Therefore the specific details of one to many, many to one, or

[6] https://en.wikipedia.org/wiki/BagIt

many to many transformations must be possible to be implemented locally on top of the common specification.

3.2. Identification of the Information Package

Requirement 2.1: *Any Information Package MUST allow for the identification of it as a SIP, AIP or DIP.*

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

Requirement 2.2: *Any Information Package MUST allow for the identification of the content type of its data.*

In the e-government context of E-ARK we use "content type" to specify the functional nature of the original business system which was used to manage the data. Examples of content types would therefore include *accounting system*, *personnel database*, *ERMS*, a more general *relational database,* or even *unstructured content*.

For E-ARK project purposes we will develop and support 4 specific "content types": ERMS; databases; geo-spatial data; and unstructured files.

However, in the wider context beyond e-government data, "content type" can also be understood more broadly as the genre of the data objects being archived, such as *publication*, *digitised images*, *digital photographs* etc. Outside E-ARK, the definition of "content type" is up to specific communities but it is important to note that content types need to be classified or organised in some way to ensure consistency in implementations.

As explained in Chapter 1.2, we expect that users of the Common Specification will start developing their own Content Type Specifications detailing the specific requirements for structure, formats and metadata of their locally used content type. As such it is crucial that metadata in an Information Package explicitly says which content types are included in the package to thus allow for the automation of in-depth validation of the package and its included content type(s).

In practice the combination of requirements 1.1, 2.1 and 2.2 allow for the development of modular identification and validation tools and workflows. While generic components can carry out high level tasks regardless of the content type, it is possible to detect automatically which additional content-aware modules need to be executed.

Requirement 2.3: *It MUST be possible to identify any Information Package uniquely within the repository.*

To be able to manage a digital repository in a reasonable manner each Information Package stored in the repository must be identified uniquely within the repository. At the same time the Common Specification must not limit the choice of the exact identification mechanism, as long as the mechanism is implemented consistently inside the repository.

Requirement 2.4: *It SHOULD be possible to identify any Information Package globally uniquely.*

In addition to the previous it is recommended that the identification mechanism implemented at the repository provides for global uniqueness of Information Packages in order to support a wider range of interoperability scenarios (for example, joining multiple repositories).

Requirement 2.5: *Any Information Package MUST, within the package, allow for the unique identification of all its components.*

Any Information Package includes multiple components – different types of metadata, data, documentation about the package etc. While the components of an IP may vary and are implementation specific, the requirement is that it is possible to uniquely identify them within the Information Package. This allows creating appropriate links between data, metadata and other components. This, in turn, ensures that it is possible in practice to maintain and validate the integrity of the package.

Note: Please look at chapte 3.3 for further requirements on the structure and components of the Information Package.

Requirement 2.6: *Any Information Package SHOULD, within the repository, allow for the unique identification of all its components.*

In addition to the previous requirement it is recommended that the uniqueness in component identification is achieved not only on the Information Package level but also within the whole repository (i.e. all components are identified uniquely across the repository). In practice this can be easily achieved when combining the unique identifier of the package (requirement 2.3 or 2.4) with the locally unique identifier of any component (requirement 2.5).

3.3. Structure of the Information Package

Requirement 3.1: *The structure of the Information Package MUST allow for the separation of data and metadata*

At the highest level each Information Package can be divided into data and metadata. It is crucial for many digital preservation tools to know explicitly which parts of the package are about data and which about metadata. If this is achieved the tools used, for example in ingest processes, can easily turn to metadata for package identification and validation tasks, and to data components for file format identification and normalisation.

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 at least into blocks of descriptive and preservation in any Information Package. In particular explicit knowledge about the location of preservation metadata will provide additional help to the appropriate tools in carrying out package validation and documentation tasks.

Requirement 3.3: *The structure of the Information Package SHOULD allow for the separation of multiple representations of data.*

Institutions might include multiple parallel representations of the same data in one Information Package, for example a database in its relational format and as a de-normalised representation. While information about such representations must be included in metadata it is also recommended that the representations are explicitly expressed in the structure of the package. This allows institutions to further simplify tasks which need to create, identify, validate or use single representations.

Requirement 3.4: *The structure of the Information Package MUST be extensible to meet additional local or business-specific needs.*

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, implementers are welcome to define these components as distinct additions to the structure of the Information Package.

Requirement 3.5: *Any Information Package MUST follow a common logical structure for its data, metadata and all other components.*

Following requirements 3.1 – 3.4 we can now state the need for a common logical structure for Information Packages. The logical structure, or data model, of the Common Specification is presented in

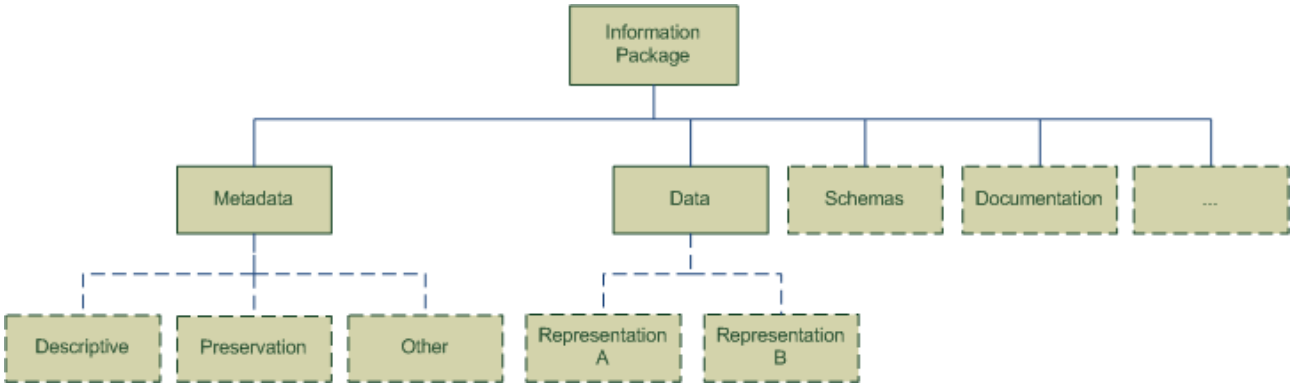


Figure 5.

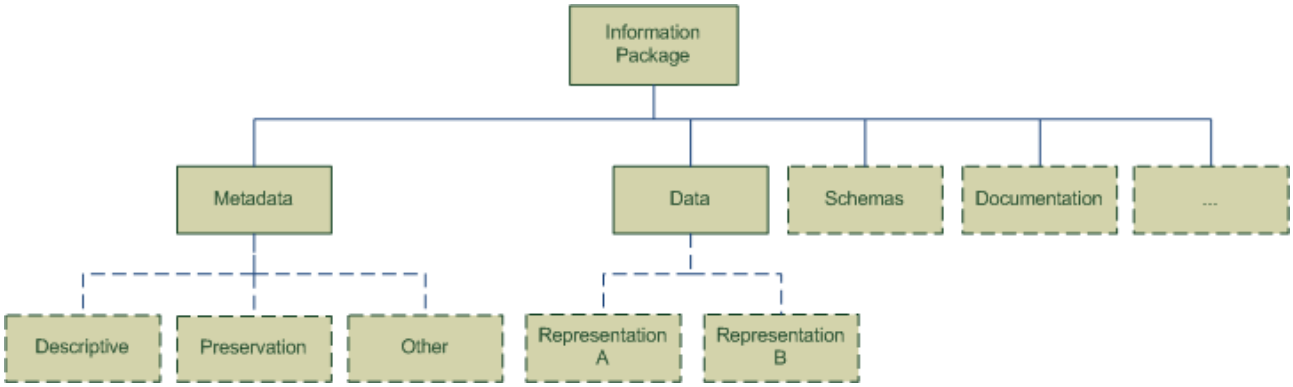


Figure 5: Conceptual structure of the Common Specification

The main characteristic of this conceptual structure is the presentation of Metadata and Data as separate and mandatory components (Requirement 3.1).

In addition we highly recommend the data model to be extended as described in the SHOULD requirements of 3.2 – 3.3 (addition of metadata sub-divisions, support for representations). These additions are also mandatory in the E-ARK implementation of the Common Specification described in the next chapters.

The data model on Figure 5 depicts also the addition of two optional components – Schemas and Documentation. However, as described in Requirement 3.4 these are not prescribed components according to the Common Specification.

Requirement 3.6: *Any Information Package SHOULD follow a common physical structure for its data, metadata, and all other components.*

The conceptual structure presented above can be implemented in various ways – the different components might be defined by accompanying package metadata or explicitly through a folder structure. Obviously, these different implementations would neither be interoperable not fulfill the purpose of the Common Specification.

Therefore, in order to achieve full interoperability, E-ARK proposes a specific physical translation of the conceptual model. This will not only enable interoperability but also allow E-ARK tools to process IPs that comply with this mandatory physical folder structure.

In chapter 4 of this document we present a physical package structure which has been developed within E-ARK and which takes into account requirements from government and business archives.

3.4. Information Package Metadata

Requirement 4.1: *Any Information Package MUST, to the largest possible extent, use internationally recognised and standardised metadata schemas for administrative, preservation, structural and technical metadata.*

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. As “crucial metadata” we

see mainly the core information about how the package has been created and managed (administrative and preservation metadata), explicit descriptions about of the structure 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 way in any Information Package, it is highly recommended to use established and widely used metadata standards. At the moment of writing, and for the foreseeable future, a large proportion of such metadata is covered by METS and PREMIS. Therefore the explicit recommendation at the moment is to adopt at least these two standards for any implementation of an Information Package.

Requirement 4.2: *Information Package metadata MUST allow validating the structure and content of any Information Package in terms of integrity, fixity and syntax.*

Most international metadata standards support multiple options for describing specific details of an Information Package. However, the availability of multiple options does make it difficult for implementers to develop common, automated and interoperable tools.

To overcome this difficulty it is required that, while devising a specific Information Package, the chosen international metadata standard is reviewed in regard to potential problems in automation and misinterpretation, and specified accordingly.

Requirement 4.3: *Any Information Package MUST allow for including 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 Type specific technical and structural metadata. To allow for the widest possible uptake of the Information Package model it has to be possible for any implementer to add whatever additional metadata is required to the mandatory metadata which are necessary for automation and interoperability.

In case organisations need to prescribe further details about descriptive or content type specific metadata for a deeper level of interoperability it is possible to use the mechanism of Content Type Specifications described above. From a more technical perspective the Common Specification therefore foresees a modular approach towards 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 types like relational databases in the SIARD2 format.

3.5. Implementation of the Common Specification

This chapter presents some final requirements which are not about the Common Specification itself but how it should be implemented in real-life scenarios. All requirements below have also been addressed in the practical E-ARK implementation described in Part II of this document.

Requirement 5.1: *Any implementation of the Common Specification SHOULD support Information Packages regardless of their size*

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.

As such 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 5.2: *Any implementation of the Common Specification 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-readable. This means that final 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 5.3: *Any implementation of the Common Specification 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 and tools are gone. 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 file and text browsers.

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 (for example, name the folder which includes package metadata "metadata").

Requirement 5.4: *Any implementation of the Common Specification MUST not limit the use of preservation methods*

Different preservation institutions need to be able to choose different methods for long-term preservation of different content types; migration and emulation being the most usual choices.

The Common Specification does not prescribe the use of a specific method. Instead it allows implementers to:

- document any preservation actions being executed on data or metadata (either as package metadata or documentation – see Requirements 3.4 and 4.2);
- include different representations of data in the Information Package (Requirement 3.3);
- add any relevant information about emulators to the package (Requirement 3.2).

PART II: E-ARK Implementation of the Common Specification

In this part of the document we describe the E-ARK implementation of the requirements and principles discussed above.

Question to reviewers: The ambition of the E-ARK project is to propose the implementation below as the de facto technical standard for Information Packages. Do you think it is fit for purpose? Do you think it is reasonable for all digital preservation scenarios (GLAM/LAM sector, private sector institutional repositories, scientific data preservation etc.) or only for public sector archives?

4. E-ARK Information Package structure

As described in Requirement 3.6, any Information Package should implement the Common Specification data model as a physical folder structure. Of course, it is in principle also possible to create an Information Package specification which does not specify a physical folder structure but instead describes the various components of the package by means of structural metadata.

On the other hand, having a fixed physical folder structure makes it clear for both human users and tools where to find what. The main benefit here is that many archival tasks (for example file format risk analysis) can be executed directly and without the need to first process 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.

One caveat with a fixed physical folder structure is that an archive might choose to use, for example, logical IP structures, or even alternative storage solutions which do not support storing data in folders. Of course this means that the specification below cannot be implemented in all situations and for IPs. However, as the main purpose of this specification is to support interoperability it is necessary to implement the fixed physical structure for package exchange scenarios (mainly SIP and DIP). Differences in actual storage (mainly relevant for AIPs) can be seen as local implementations and not relevant for achieving interoperability.

Based on these considerations the E-ARK project has decided to include the requirement of a fixed folder structure into its implementation. In other words – all tools developed and piloted in the project will follow the structure described below.

4.1. Basic structure of the E-ARK Information Package

The basic E-ARK Information Package folder structure is presented in Figure 6 below. The structure follows directly the principles of the Common Specification requirements and the logical data model by dividing the components of the package into stand-alone folders.

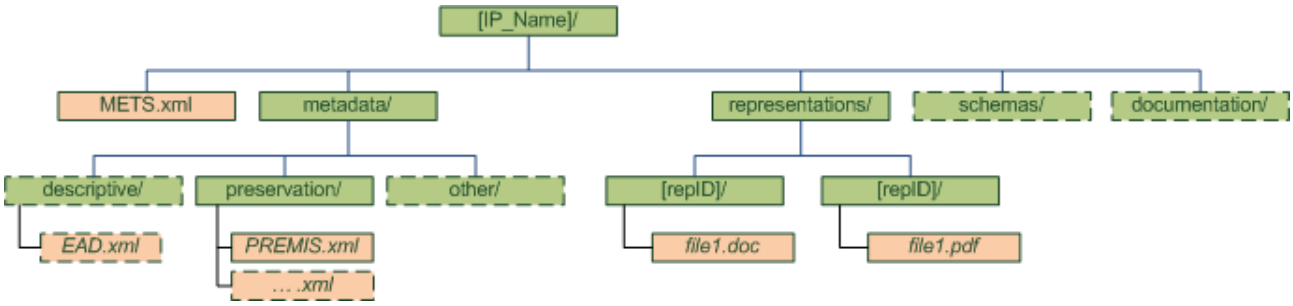


Figure 6: Basic E-ARK Information Package folder structure

The main characteristics and requirements of the basic E-ARK Information Package structure are:

- Each E-ARK Information Package *must* be included in a single container (folder, compressed file or similar) named with the ID or name of the Information Package;
- The Information Package folder *must* include a mandatory core metadata file called **"METS.xml"**, which includes core information needed to identify and describe the structure of the package itself and the rest of its components[7].
- All other metadata *must* be placed into the folder called **"metadata"**
    - All preservation metadata *must* be included into the mandatory sub-folder **"preservation"**. This folder must include administrative, technical and preservation metadata in one or many files, most typically in PREMIS format;
    - If descriptive metadata are available, they *must* be included in the sub-folder **"descriptive"**;
    - If any other metadata are available, they *can* be included into separate sub-folders, for example an additional folder named "other".
- All data *must* be included in the folder **"representations"**;
    - The "representations" folder *must* include a sub-folder for each individual representation named with a string uniquely identifying the representation within the scope of the package. This rule is also valid in the case of including only one representation into the IP - implementers are not allowed to skip the single representation folder or the "representations" folder[8];
    - The sub-folder of a representation can include content files directly or also introduce further folder hierarchies;
- The E-ARK recommendation is to also include in the IP all XML Schemas used for the package. All these schemas *should* be placed into the folder called **"schemas"**.
  Note that this is not a mandatory requirement and institutions might choose to not make use of the "schemas" folder and instead reference XML Schemas externally or keep the schemas together with the XML files in metadata folders. All these alternative approaches will still result in a valid package structure;

[7] For a detailed description of the content of the METS.xml file please see chapter 5.

[8] 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 ensure consistent implementation the requirement to include a representation folder remains intact.

- The folder structure can also be extended to meet specific national or local needs. As an example, implementers can add a ”documentation” folder next to the mandatory folders in order to gather all binary documentation about the data or the Information Package. Those additional components of the physical structure should always be seen as extensions to the E-ARK Information Package specification and must not replace or change any of the above-mentioned folders.

4.2. Full structure of the E-ARK Information Package

According to our understanding the structure presented in chapter 4.1 is sufficient for most scenarios and packages. However, the concern with such a simple structure is scalability. In the case of multiple representations and extensive amounts of data (for example, a package including three parallel 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. Therefore, the E-ARK project proposes in addition an extended, more scalable and flexible, folder structure as depicted in Figure 7.

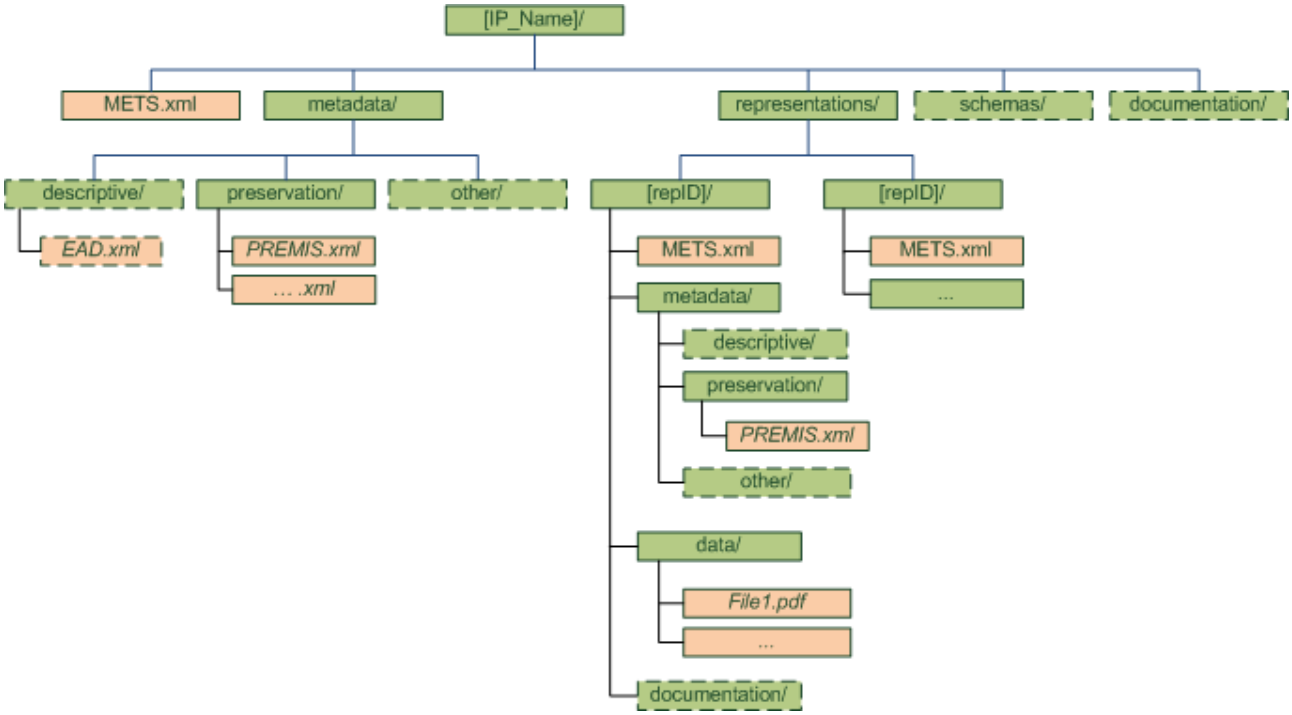


Figure 7: Full E-ARK Information Package folder structure

The requirements posed for the simple structure remain valid. The main difference between the simple and full structure is that each representation in the full structure does not only include the data files but repeats the simple structure. Each representation in the full structure is thus the equivalent of the simple structure. As such the full structure gives institutions the possibility of dividing metadata between the root level of the package and the representations, making the whole package easier to manage and single representations easier to understand and reuse.

The additional requirements for the full structure are as follows:

- Use of the METS.xml file
    - The METS.xml file in the root of the package includes information about the full package and lists the representations, but not the components of the representations;
    - The METS.xml file in each of the representations includes full information about the components of the exact representation, including details about the data;
- Use of the "metadata" folder
    - Root level "metadata" folder and its sub-folders should only include metadata relevant to the whole package (most typically descriptive metadata and preservation metadata relevant for the whole package);
    - "metadata" folders within a representation should only include metadata which is relevant for that representation (most typically preservation metadata about the creation and management of the representation and technical metadata for any computer files);
- Content files must be placed into a sub-folder called **"data"**;
- Use of "schemas" folder
    - E-ARK recommends using the "schemas" folders in a similar way to the "metadata folder". The optional "schemas" folder within a representation should include only the XML Schemas which are appropriate for the specific representation. All schemas which are used across multiple representations and/or in the root metadata folder should be placed into the "schemas" folder in the root of the package.
      In addition we recommend avoiding the duplication of schemas across different folders.
- Institutions are welcome to add any further folders either into the root level of the package or into representations (for example a "documentation" folder). In those cases the universal rule is that all components need to be placed at the lowest reasonable level (i.e. information about the full package into the root level and information about a single representation into the representation level).

Question to reviewers: All E-ARK tools have been developed to support both the simple and full Information Package structure in an interchangeable way. Do you think it would be possible and reasonable to stick only to one of these in further E-ARK work?

5. Use of metadata

5.1. General requirements for metadata in an E-ARK Information Package

Some of the core metadata requirements are already visible from the structure presented in the previous chapter. Most crucially the E-ARK Information Package requires a minimal set of metadata elements and that institutions divide different types of metadata into separate folders of the package.

Central components of the E-ARK Information Package are the METS files. As described above, all Information Packages need to include one and only one METS file in the **root** of the package, named "METS.xml". In case the full structure has been implemented, the package needs to include one additional "METS.xml" file in the root of each representation. These files will be referred to as "root METS" and "representation METS" in the rest of this document[9].

As well as the METS files any E-ARK Information Package needs to include PREMIS metadata in appropriate preservation metadata folders. The naming and number of these 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 ones. The only requirement is that all PREMIS files must be listed and referenced from appropriate METS files, i.e. root PREMIS files from the root METS file and representation PREMIS files from the representation METS files.

The use of any additional metadata is not restricted in E-ARK Information Packages, allowing institutions to include any additional metadata into the package. However, please note that additional restrictions especially in regard to descriptive metadata can be specified in the Content Type Specifications.

5.2. Use of METS in an E-ARK Information Package

The main requirement for METS files in an E-ARK Information Package is that these need to follow the official METS Schema version 1.11[10,11].

The following text assumes knowledge of the principles of the METS specifications. If this is not the case, please consult the official documentation[12] before continuing.

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

[9] Please note that the current version of this specification describes only the METS files for the full physical IP structure. The METS file included in the simple IP structure will be a composition of the root and representation METS files. A detailed description of such a "simple METS" will follow in the next version of this specification.
[10] Available at http://www.loc.gov/standards/mets/version111/mets.xsd
[11] The E-ARK project will establish a sustainable update and management regime for the Information Package structure by 2017. This will also include the monitoring of any future updates to the METS schema and appropriate modifications.
[12] Available at http://www.loc.gov/standards/mets/mets-schemadocs.html

Use of the METS root element (<mets>)

The purpose of the METS root element is to establish 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.

The *xsi:schemaLocation*[13] attribute of the METS root element *<mets>* must refer to all necessary XML schemas. In the case of the recommended use of the "schemas" folder all schemas need to be referred by relative path (as an example: "schemas/mets.xsd" in the case of the root METS.xml file and "../../schemas/mets.xsd" in the case of the representation METS.xsd file).

The specific requirements for the root element and its attributes are described in the following table[14].

Element name	*Attribute name*	*Use in root METS.xml*	*Use in representation METS.xml*
<mets>	ID	Optional, no further requirements	
	OBJID	Mandatory. Must 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	Mandatory. Must be the same as the ID of the representation (the name of the representation folder). The OBJID must meet the Common Specification requirement of being unique at least within the package
	LABEL	Optional, if used should be filled with a human-readable description of the package	Optional, if used should be filled with a human-readable description of the representation
	TYPE	Mandatory. The TYPE attribute must be used for identifying the OAIS type of the package (i.e. SIP, AIP, DIP, AIC) and the content type of the package (ERMS, RDBMS, other, mixed). The value has to be expressed according to the following rule: *<OAIStype>*:*<ContentType>.* Example: "*SIP:database*" Please note that the next version of the E-ARK IP specification will include specific vocabularies for the values of the TYPE attribute	Mandatory. The TYPE attribute must be used in a similar way as for the root METS file with the exception that instead of the OAIS type the first part of the attributes value is a fixed string "representation". Example: "representation:database"
	PROFILE	Mandatory. The PROFILE attribute has to be filled with the URL of the official E-ARK METS Profile. As this is not yet available	Not used

[13] xsi stands here for the common namespace prefix of the schema at URL http://www.w3.org/2001/XMLSchema-instance

[14] 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.

		the placeholder value to be used is "http://www.eark-project.com/METS/IP.xml"	

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="AIP:database" OBJID="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">
```

Use of the METS header (<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.

Element name	*Attribute name*	*Use in root METS.xml*	*Use in representation METS.xml*
metsHdr	ID	Optional, no further requirements	
	ADMID	Optional, referring to the appropriate administrative metadata section if available	
	RECORDSTATUS	Optional, no further requirements	
	CREATEDATE	Mandatory, the date of creation of the package	Mandatory, the date creation of the representation
	LASTMODDATE	Mandatory if relevant (in case the package has been modified)	Mandatory if relevant (in case the representation has been modified
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.	Optional, no further requirements
metsHdr/altRecordID	ID	Optional, no further requirements	
	TYPE	Optional, no further requirements	
metsHdr/ metsDocumentID		Optional, E-ARK recommends the value to be the same as OBJID.	

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>
```

Use of the METS administrative metadata section (<amdSec>)

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

Due to the Common Specification requirement 3.2 (any Information Package should separate different types of metadata) all preservation metadata must be stored outside of the METS.xml file and referenced by using the <mdRef> element and thus not embedded (i.e. the use of <mdWrap> element is not allowed).

The METS <amdSec> element must include references to all relevant metadata 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 E-ARK Information Package requires having all administrative metadata described in a single <amdSec> element (i.e. not repeatable).

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

Element name	*Attribute name*	*Use in root METS.xml*	*Use in representation METS.xml*
amdSec		Mandatory to include exactly one <amdSec> which refers to all root level preservation metadata files	Mandatory to include exactly one <amdSec> element which refers to all representation level preservation metadata files
	ID	Mandatory, identifier must be unique within the package	
amdSec/digiprovMD		Mandatory to include one <digiprovMD> element for each file in the "metadata/preservation" folder.	
	ID	Mandatory, identifier must be unique within the package	
	GROUPID	Optional, no further requirements	
	ADMID	Optional, no further requirements	
	CREATED	Not used	
	STATUS	Mandatory, must include one of the two values [superseded, current]	
amdSec/digiprovMD /mdWrap		Not used	
amdSec/digiprovMD		All references to the metadata files should be made using the	

/mdRef		XLink href attribute and the file protocol using the relative location of the file. Example: `xlink:href="file:metadata/preservation/premis.xml"` This requires, in turn, the usage of the XLink type attribute with the value “simple”. Example: `xlink:type="simple"`
	ID	Mandatory, identifier must be unique within the package
	LOCTYPE	Mandatory, always using value “URL”
	CREATED	Mandatory, used according to the official METS guidelines
	CHECKSUM	Mandatory, used according to the official METS guidelines
	CHECKSUMTYPE	Mandatory, used according to the vocabulary presented in the official METS schema
	MDTYPE	Mandatory, used according to the vocabulary presented in the official METS schema
	MIMETYPE	Mandatory, used according to the official METS guidelines
amdSec/techMD		The use of <techMD> is not recommended. Instead, detailed technical metadata should be included into or referenced from appropriate PREMIS files
amdSec/rightsMD		Optional. E-ARK recommends including a simple rights statement which describes the overall access status of the package (as an example with values: *open, closed, partially closed, not known*). However, the exact schema and element is up to individual implementations to decide
amdSec/sourceMD		Optional, no further requirements

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>
```

Use of the METS descriptive metadata section (<dmdSec>)

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

The use of the <dmdSec> element follows the same rules as <amdSec>: all descriptive metadata must be placed as separate files into the metadata/descriptive folder and referenced using the <mdRef> element.

The attributes of the <dmdSec> and the <mdRef> elements must be used according to the same requirements as provided for the <digiprovMD> above.

If the package includes multiple versions of the same metadata (as an example an EAD file created by the submitting entity and another version updated by the archives) these must be presented as separate <dmdSec> occurrences. In this case we also recommend using the STATUS attribute of the <dmdSec> element with values “current” or “superseded”.

Full example of the METS <dmdSec> element:

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

Use of the METS file section (<fileSec>)

Use of the METS <fileSec> element is highly recommended by E-ARK (though not mandatory). It should describe all files within the package which have not been 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” (the latter is the term that E-ARK is using) and allow validating the integrity of the files included into the package.

The main requirement of the E-ARK IP specification is that the file section (<fileSec> element) of both the root and representation METS files include at least one file group (<fileGrp> element). This so-called “E-ARK 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 “E-ARK files root” (for the root METS file) or “E-ARK files representation [representation ID]” (for the representation METS file)
    - *Example: <fileGrp USE=”E-ARK files 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”);
- The 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 describe the representation METS file in it;

The specific requirements for elements, sub-elements and attributes are listed in the following table:

Element name	*Attribute name*	*Use in root METS.xml*	*Use in representation METS.xml*
fileSec		Recommended to include one <fileSec> element into each METS file	

fileSec/fileGrp		Recommended to include one E-ARK defined <fileGrp> element. Implementers are welcome to define and add additional file groups necessary for internal purposes.	
	ID	Mandatory, identifier must be unique within the package	
	USE	Mandatory, value must be "E-ARK files root"	Mandatory, value must be "E-ARK files representation [representation ID]
fileSec/fileGrp/ fileGrp/…		The main <fileGrp> element includes additional nested <fileGrp> elements, one for each folder of the package (except metadata described in <amdSec> and <dmdSec>)	
	ID	Mandatory, identifier must be unique within the package	
	USE	Mandatory, value must be the same as the name of the folder (schemas, documentation, data, etc)	
fileSec/fileGrp/…/file		Each file within the folders described by <fileGrp> elements by one <file> element	
	MIMETYPE	Mandatory	
	USE	Optional, no further requirements	
	CHECKSUMTYPE	Mandatory, values according to the official METS guidelines	
	CREATED	Mandatory	
	CHECKSUM	Mandatory	
	ID	Mandatory, must be unique across the package	
	SIZE	Mandatory	
fileSec/fileGrp/…/ file/FLocat		The location of each file must be defined by the <FLocat> element using the same rules as for referencing to 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"` 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"	

Example of the <fileSec> element (root METS file):

```
<fileSec>
   <fileGrp USE="E-ARK files 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="E-ARK files 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/contentfile.pdf" xlink:type="simple"
LOCTYPE="URL"/>
            </file>
         </fileGrp>
         ...
      </fileGrp>
      <fileGrp USE="documentation" ID="ID7d136e4c-26fe-40da-85a2-67a42efd6b27">
      ...
      </fileGrp>
   </fileGrp>
</fileSec>
```

Use of the METS structural map (<structMap>)

The purpose of the METS structural map section is to provide an overview of ALL components of an E-ARK 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, E-ARK tools 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 E-ARK Information Package and must always be present.

The E-ARK 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. These additional structural maps are not exploited in E-ARK tools.

The most crucial requirements for the E-ARK mandated structural map are as follows:

- The <structMap> element has a mandatory attribute LABEL which has the fixed value of "E-ARK structural map". The LABEL attribute is used to distinguish the E-ARK structural map 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 "E-ARK structural map";
- The internal structure of the structural map (expressed by hierarchical elements) follows the E-ARK physical structure as described in chapter 4, therefore grouping together metadata, representations, schemas, documentation and user-defined folders;
    - All elements must use the attribute LABEL with the value being the name of the folder (as an example "metadata")
- 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 elements)
    - Lists only the appropriate METS file using the <mptr> element as the content of the representation
- The structural map in the 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:

<table>
<tr><th>Element name</th><th>Attribute name</th><th>Use in root METS.xml</th><th>Use in representation METS.xml</th></tr>
<tr><td>structMap</td><td></td><td colspan="2">Each METS file needs to include exactly one <structMap> element which is used exactly as described in this table. Institutions can add their own custom structural maps as separate <structMap> elements next to it</td></tr>
<tr><td></td><td>ID</td><td colspan="2">Optional, if used must be unique within the package</td></tr>
<tr><td></td><td>TYPE</td><td colspan="2">Mandatory, value must be "physical"</td></tr>
<tr><td></td><td>LABEL</td><td colspan="2">Mandatory, value must be "E-ARK structural map"</td></tr>
<tr><td>structMap/div</td><td></td><td colspan="2">Each folder (and sub-folder) within the package must be represented by an occurrence of the element. Please note that sub-folders must be represented as nested div elements.

Example:

<structMap TYPE="physical" LABEL="E-ARK structural map">

...

</structMap></td></tr>
<tr><td></td><td>ID</td><td colspan="2">Mandatory, identifier must be unique within the package</td></tr>
<tr><td></td><td>ORDER</td><td colspan="2">Not used</td></tr>
<tr><td></td><td>ORDERLABEL</td><td colspan="2">Not used</td></tr>
<tr><td></td><td>LABEL</td><td>Mandatory, value must be the name of the folder („metadata“, „descriptive“,</td><td>Mandatory, value must be the name of the folder („metadata“, „descriptive“,</td></tr>
</table>

		„schemas“, „representations“, etc). The LABEL value of the first element in the package is the ID of the package	„schemas“, „data“, etc). The LABEL value of the first element in the package is the ID/name of the representation
	DMDID	No specific requirements	
	ADMID	No specific requirements	
	TYPE	No specific requirements	
structMap/div/…/div/ fptr		If the folder which is described by the 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.	Inside the representation METS file <fptr> element is used to reference all files within the representation with no exceptions. The <fptr> child elements <par>, <seq> and <area> must not be used.
	ID	No specific requirements	
	FILEID	Mandatory, must be the ID used in the appropriate <file> or <mdRef> element	
	CONTENTIDS	No specific requirements	
structMap/div/div/ mptr		In the case of describing representations within the package (i.e. 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”.	Not used

		Example: `xlink:type="simple"` The LOCTYPE attribute is used with the fixed value ”URL”	
	ID	Not used???	
	CONTENTIDS	Not used???	

Full example of the E-ARK <structMap> element (root METS file):

```
<structMap TYPE="physical" LABEL="E-ARK 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>
</structMap>
```

Use of the METS Structural Link Section (<structLink>) and Behavior Section (behaviorSec)

The E-ARK Information Package implementation poses no additional requirements on the METS <structLink> and <behaviorSec> elements.

5.3. Use of PREMIS in an E-ARK Information Package

The main requirement of the E-ARK Information Package is that preservation metadata are being recorded into the package in PREMIS format. Accordingly all E-ARK tools are expected to create, and be able to validate and make use of, PREMIS metadata.

However, the interoperability scope of the E-ARK Information Package does not require imposing any further detailed restrictions as is the case with the use of METS. Instead it is sufficient if:

- All preservation metadata is created according to official PREMIS guidelines[15];
- All PREMIS files are described and referenced in the <amdSec> element of the appropriate METS file;

There are just a few best-practice recommendations which will be followed in E-ARK tools and are also recommended to any external tools and implementers:

- Information about agents carrying out preservation actions should be recorded in PREMIS. The use of METS agents should be limited to these agents which are relevant for generic IP level events (for example, the creation of the package, submitting agency);
- Event descriptions should be included into PREMIS metadata to the largest possible extent. We recommend using the official PREMIS event vocabulary[16];
- Detailed rights information should be available 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[17]. In case high level rights information in METS highlights restrictions tools or humans can start looking at detailed, object-specific, rights information in PREMIS;
- PREMIS should be used to record detailed technical metadata. In METS the level of technical metadata should only include the checksums and size of files;
- As much technical metadata as possible should be included in PREMIS files by using extension schemas;
- We recommend adding file format information as PUID[18] values for all files into appropriate PREMIS metadata.

[15] http://www.loc.gov/standards/premis/
[16] http://id.loc.gov/vocabulary/preservation/eventType.html
[17] Cf. Chapter "Use of the METS administrative metadata section (<amdSec>)"
[18] PUID stands here for "PRONOM Persistent Unique Identifier". See http://www.nationalarchives.gov.uk/PRONOM for more information.

6. Next steps on the Common Specification

This document is a first public draft of the Common Specification and the overall E-ARK Information Package specification. This draft has been prepared with the goal to attract further opinions and comments. **We would appreciate to receive your comments by the end of February 2016!**

Based on the external comments as well as internal developments the E-ARK project will continue updating and adjusting the specifications. The overall goal of the project is that by the end of E-ARK (January 2017) the Common Specification is sufficiently mature to be presented for international standardisation as well as uptake by major industry players.

Some of the known next steps and additions to the Common Specification include:

- A clearer separation and explanation of the two physical structures (simple and full). We will especially aim to research and clarify the use of METS in the case of the simple physical structure;
- Introduction of the "package of packages" concept which would allow to split large information packages into multiple physical packages and describes these splits by a METS file similar to the one described in this document;
- Further explanations about the use and creation of Content Type Specifications;
- Full examples of the E-ARK Information Packages. For now examples are maintained on GitHub (https://github.com/eark-project/information-package/tree/master/examples). However, these are not final but "work in progress" and thus being changed regularly.