SLOVENSKI STANDARD
SIST-TP CEN/TR 17862:2022

01-oktober-2022

Smernice za izvajanje paketa za varstvo filmov (CPP) v EN 17650

Guideline for the implementation of the Cinema Preservation Package (CPP) in EN 17650

Leitlinien für die Umsetzung des Cinema Preservation Package (CPP) in EN 17650

Lignes directrices relatives à la mise en œuvre du paquetage de conservation cinéma (CPP) dans l'EN 17650

Ta slovenski standard je istoveten z: **CEN/TR 17862:2022**

ICS:

35.240.30	Uporabniške rešitve IT v informatiki, dokumentiranju in založništvu	IT applications in information, documentation and publishing
37.060.99	Drugi standardi v zvezi s kinematografijo	Other standards related to cinematography

SIST-TP CEN/TR 17862:2022 **en,fr,de**

TECHNICAL REPORT

RAPPORT TECHNIQUE

TECHNISCHER REPORT

CEN/TR 17862

August 2022

ICS 35.240.30; 37.060.99

English Version

Guideline for the implementation of the Cinema Preservation Package (CPP) in EN 17650

Lignes directrices relatives à la mise en œuvre du paquetage de conservation cinéma (CPP) dans l'EN 17650

Leitlinien für die Umsetzung des Cinema Preservation Package (CPP) in EN 17650

This Technical Report was approved by CEN on 17 July 2022. It has been drawn up by the Technical Committee CEN/TC 457.

CEN members are the national standards bodies of Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Republic of North Macedonia, Romania, Serbia, Slovakia, Slovenia, Spain, Sweden, Switzerland, Türkiye and United Kingdom.

EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

CEN-CENELEC Management Centre: Rue de la Science 23, B-1040 Brussels

Ref. No. CEN/TR 17862:2022 E

Contents

Page

European foreword

This document (CEN/TR 17862:2022) has been prepared by Technical Committee CEN/TC 457 “Digital preservation of cinematographic works”, the secretariat of which is held by DIN.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CEN shall not be held responsible for identifying any or all such patent rights.

This document has been prepared based on the Rolling Plan for ICT standardization 2015-2018.

This document serves as implementation guide to the European Standard EN 17650: *A framework for digital preservation of cinematographic works — The Cinema Preservation Package*.

Any feedback and questions on this document should be directed to the users’ national standards body. A complete listing of these bodies can be found on the CEN website.

Introduction

This document is part of a series of standards and technical recommendations for the digital preservation of cinematographic work. It gives European film archives and producers a guideline how to store and manage film content in the digital age.

Versions of digital cinematographic work using different encoding formats can be preserved in a layered structure where the lowest level is describing the physical file. The files can carry data representing images, sound, movies, subtitles, metadata or ancillary information like QC reports or film posters.

To structure these files in a consistent and interoperable way, the Cinema Preservation Package (CPP) is specified and adds additional metadata like file lists, links or hash values. The Cinema Preservation Package uses different XML files to store these additional metadata. The XML files are based on existing schemas with extensions, in particular METS, EBUCore and PREMIS. The XML files also contain hash values to ensure data integrity and version control. The syntax for this description and the methods for the hash value generation are specified in EN 17650. Various types of content coding are described as reference for concrete implementations.

The Cinema Preservation Package is well suited to serve as a Submission Information Package (SIP) in an OAIS compliant preservation system [2] and/or as a self-contained exchange format between media archives. The CPP does not necessarily contain a complete cinematographic work, it can also be used for exchange of parts of a work (see Annex G).

1 Scope

This document describes the implementation of the Cinema Preservation Package (CPP) to facilitate the digital preservation of cinematographic works. The related standard EN 17650 specifies methods to describe the relationship of components of the cinematographic work and delivers syntax to describe the package content. EN 17650 specifies the structure of the package and the constraints that are necessary to enable compliance and interoperability. This document gives additional information how to implement EN 17650 and gives additional explanations to the structure.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

EN 17650, *A framework for digital preservation of cinematographic works — The Cinema Preservation Package*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in EN 17650 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- ISO Online browsing platform: available at http://www.iso.org/obp
- IEC Electropedia: available at http://www.electropedia.org/

3.1
ancillary data
small additional data supporting other data in the same hierarchy, preferably annotations

3.2
element
simple or essential part of which anything exists

3.3
file
collection of information stored on a digital storage medium

3.4
persistent identifier
unique identifier that ensures permanent access for a digital object by providing access to it independently of its physical location or current ownership

3.5
preservation
act of maintaining information in a correct and independently understandable form over a long time

4 Abbreviations

ACES Academy Color Encoding System

ASCII American Standard Code for Information Interchange

BMFF	Base Media File format
CPP	Cinema Preservation Package
DCP	Digital Cinema Package
DPX	Digital Picture Exchange
IMF	Interoperable Master Format
JPEG	Joint Photographic Experts Group
LoC	Library of Congress, see https://www.loc.gov/
MPEG	Moving Pictures Experts Group
OAIS	Open Archive Information System
PCM	Pulse code modulation
PDF	Portable Document Format
PNG	Portable Network Graphics
SIP	Submission Information Package
TIFF	Tagged Image File Format
UUID	Universal Unique Identifier
XML	Extensible Markup Language

5 Syntax convention used in this document

5.1 General

For the file and folder name descriptions the following conventions are used.

5.2 Expressions used to denote file or folder names

5.2.1 Composition of name with parts

A name may be composed of one or several parts.

5.2.2 Literal parts

A literal part appearing between quotes appears as is.

EXAMPLE “LiteralPart” designates a part which is exactly “LiteralPart”.

5.2.3 Optional part

A part appearing between brackets appears zero or one time.

EXAMPLE [“OptionalPart”]”Radical” designates a name which is either “OptionalPartRadical” or “Radical”.

5.2.4 Alternative parts

Exactly one of the items in parentheses, separated by a vertical line ‘|’, appears in the result.

EXAMPLE (“Prefix1”|“Prefix2”|“Prefix3”)“Radical” designates a name which is either “Prefix1Radical”, “Prefix2Radical” or “Prefix3Radical”.

5.2.5 Explicitly defined part

<any text here> is a placeholder text and is replaced as specified by the prose.

EXAMPLE "Name_"*<custom identifier>* designates a name which could be "Name_0123456789", supposing "0123456789" is defined as a valid custom identifier.

5.3 Expressions used to denote file or folder multiplicity

5.3.1 General

Some files or folders may appear several times in the parent folder. The final modifier therefore indicates the allowed multiplicity.

5.3.2 One occurrence or more

A file or folder name with a trailing '+' appears one or more times in the parent folder.

EXAMPLE "File_"*<number>*+ designates a set of multiple files such as "File_0000", or "File_0000" and "File_0001" and "File_0002", etc., assuming the prose describes such a 4-digit numbering.

5.3.3 Zero occurrence or more

A file or folder name with a trailing '*' appears zero or more times in the parent folder.

EXAMPLE "File_"*<number>** designates a set of multiple files such as no file, "File_0000", or "File_0000" and "File_0001" and "File_0002", etc., assuming the prose describes such a 4-digit numbering.

5.3.4 Optional file or folder

A file or folder name with a trailing '?' appears zero or one time in the parent folder.

EXAMPLE "OptionalFile"? designates no file, or exactly one file with name "OptionalFile".

5.4 Typographic conventions

5.4.1 Monospaced fonts

In the document, monospaced fonts are used for literal expression of textual content of a file, for instance as example quotation.

EXAMPLE <exampleXmlElement>example text</exampleXmlElement>

5.4.2 Italic

In the document, italic text is expected to be substituted as specified in the prose.

EXAMPLE *relative path to file* is the path from the root folder to the Descriptive Metadata file including the filename.

5.5 Conventions used to denote XML content

5.5.1 General consideration

Constraints on XML files are specified either as an XML Schema to apply, or as listed constraints for each element of a list of specific XML Nodes.

5.5.2 Hierarchy

To denote the hierarchical position of a node the XPath syntax is used.

EXAMPLE "/rootElement/subElement/subSubElement" denotes an element with name "<subSubElement>" which is a child of a "<subElement>" element, which itself is a child of the "<rootElement>" root element.

NOTE When an element in the path occurs more than one time, the XPath will be completed with discriminary attributes if necessary. If the description applies to multiple elements, no discriminatory attribute will be added.

5.5.3 Namespace

XML elements may use a prefix to denote a namespace. The prefix indicated in the prose is only used for convenience, and does not denote a namespace. The prefix in the file may be different and this is suitable as long as the relation with the namespace is correctly indicated as a namespace attribute from a parent element.

EXAMPLE "/mainNs:rootElement/mainNs:subElement" the "<rootElement>" and the "<subElement>" belong to the namespace denoted by "mainNs" in the prose.

5.5.4 Subelements

The following are used:

— @attributeName: denotes the name of an attribute of the element;

— #children: denotes the child elements;

— #text: denotes the child text of the element.

EXAMPLE

```
<mainNs:rootElement attribute1="example 1">
    <subElement1/>
    <subElement2>example 2</subElement2>
</mainNs:rootElement>
```


"/mainNs:rootElement/#children" denotes "subElement1" or "subElement2"

"/mainNs:rootElement/@attribute1" denotes "attribute1" with value "example 1"

"/mainNs:rootElement/subElement2/#text" denotes "example 2"

5.5.5 Property

If the XPath includes a part inside brackets, this is a constraint to properly address the right element.

EXAMPLE "/mainNs:rootElement/subElement[@position='first']" denotes the <subElement> element which is a child of the root element <rootElement> and has an attribute "position" with value "first".

5.5.6 Conventions used in XML constraints list tables

Prior to each table, the current XML element is reminded using the XPath syntax.

In the following sections, the presence of elements is specified using tables. In these tables the "Element Arity" column indicates the multiplicity allowed for an element:

- 1: the element is mandatory, exactly one element is present;
- 0/1: the element is optional, it appears zero or one time;
- 1/n or 1/∞: multiple mandatory. The element appears at least one time and may be repeated;

- 0/n or 0/∞: multiple optional. The element might not be present, or might be present one or multiple times.

For each element subelements are indicated. The “Subelement mandatory” column indicates if this subelement is mandatory. An ‘x’ in the corresponding cell indicates that the subelement is present, an ‘o’ in the same column indicates the subelement may be present. An attribute starts with the character ‘@’. #text denotes the child text, if mandatory, the element cannot be empty. #children indicates xml child elements.

Aiming to reduce the number of tables and sections in this document, when the semantic allows to do so consistently, subelements are sometimes documented in the same table when only few elements are described.

EXAMPLE ‘title/dc:title’ indicates that the subelement `<title>` contains itself a sub-subelement `<dc:title>`, meaning that the whole XML segment has to be added to the current element described.

```
<title typeLabel="titleType">
  <dc:title lang="fr"> title name</dc:title>
</title>
```

6 Usage of XML schemas in CPP

The CPP incorporates the specifications METS, EBUCore and PREMIS and its related XML schemas. The specifications will be used separately in different files for specific utilizations in the CPP. The schemas of the specifications will not be mixed in one file.

Each of these recommendations has some few mandatory requirements to conform with; in addition, they provide much more optional elements allowing adjusting each of these recommendations to specific client use cases.

CPP extends EBUCore specification with the EBUCore internal extension mechanism using technicalAttributes. CPP do not use METS extension points. CPP does not prohibit the use of METS extension points. CPP does not extend PREMIS. A minimal CPP does not require any PREMIS document at all.

For achieving highest interoperability the extensions for a specific schema should be limited to the extensions provided in the standard EN 17650. CPP compliant does not automatically mean that the full set of METS, EBUCore and PREMIS elements must be supported, but the minimal set of mandatory elements has to be met for all recommendations.

Interoperability differs from standard conformance. To maximize interoperability your CPP tool-set should support all mandatory and optional elements of METS, EBUCore and PREMIS.

Annex A
(informative)

Structure of the Cinema Preservation Package

A.1 General

This annex describes the structure of the Cinema Preservation Package (CPP) in general parlance and with illustrative drawings and gives advice for implementing the package. The CPP is aimed to contain parts or all elements of a cinematographic work. In the case where only parts of a cinematographic work are contained, the CPP can be used as exchange format for a film scan or a specific variant between archives or between service provider and archives. In other use cases the CPP can be used as long-term preservation format or as submission format to an archive system.

A.2 CPP overview

The Cinema Preservation Package follows a specific physical structure as illustrated in Figure A.1.

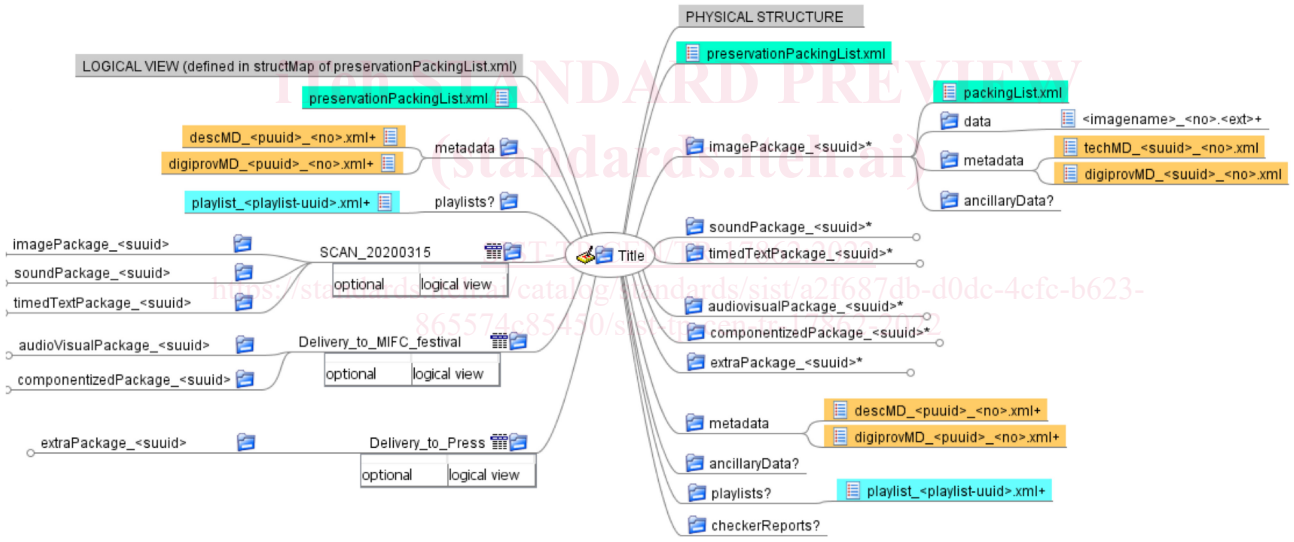

Figure A.1 — Physical structure of a CPP

The basic elements are:

- Packing lists, denoted as **preservationPackingList.xml** or **packingList.xml**, which list elements in subfolders, link audiovisual content files to metadata files and include hash values for the listed files.
- Content in Subpackages or data folders: for the root level, content is stored as subpackages. For the subpackages, content files are stored in its data folder, denoted as **data.** Each subpackage contains content files of only one specific content type.
- Metadata folders, denoted as **metadata**, for metadata files. In the subpackage these metadata are specific to the subpackage data. The metadata folder on the root level contains metadata that are applicable to the complete CPP.

- Ancillary data folders, denoted as **ancillaryData**, for ancillary data files which are neither audiovisual content nor metadata but can help in interpreting the audiovisual data files, so called annotation files. In the subpackage ancillary data files are specific to the subpackage data. The ancillary data folder on the root level contains ancillary data files that are applicable to the complete cinematographic work.

In addition to the basic elements the CPP can contain in the root folder a playlists folder, denoted as **playlists**, with files for linking a group of audiovisual content files to another group of audiovisual content files located in a different subpackage or a checker reports folder, denoted as **checkerReports**, for check report files of the package.

A.3 Subpackage

A.3.1 Introduction

The Cinema Preservation Package is organized in subpackages that will contain movie data or extra data in part or as a whole. Each subpackage contains **exactly one type of audiovisual data** or **for general purposes extra data,** together with supporting data (metadata or ancillary data). The type of audiovisual data can be image sequences, sound files, timed-text data, wrapped movie files like MP4 files or multichannel sound files, or complete componentized packages like DCPs or IMF packages. Extra data subpackages contain data elements associated with the cinematographic work like press kits, poster or archive specific data records.

All subpackages with movie data contain one metadata file in the metadata folder for technical information about the audiovisual data and optionally metadata files for digital provenance, events and processes. In addition, subpackages can contain ancillary data files in the ancillary data folder, typically notes or single sheets (annotations), associated to this specific subpackage.

A.3.2 Basic structure

The basic structure of a subpackage is illustrated in Figure A.2.

Figure A.2 — Subpackage structure

A.3.3 Subpackage folder

All movie data belonging to one consecutive piece of the movie or extra data to the cinematographic work are contained in a subpackage. The subpackage is stored in a subfolder called `<subpackagetype>_<suuid>`. Thus, movie data or extra data can be uniquely identified by its type and its unique identifier.

The subpackage types are given in Table A.1.

Table A.1 — Subpackage types

<subpackagetype>	Description
imagePackage	This subpackage type contains one sequence of image files, e.g. *.dpx, *.tif or *.exr, which are temporally consecutive in the movie (typically one reel).
soundPackage	This subpackage type contains one set of single channel sound files, e.g. *.wav, which are temporally associated (typically one reel).
timedTextPackage	This subpackage type contains timedText files, e.g. *.xml or *.png files, which are temporally associated (typically one reel).
audiovisualPackage	This subpackage type contains movie data or multichannel sound in wrapped files, e.g. *.mp4, *.mov, *.mkv or *.wav. The wrapped files form a playable piece of the movie or sound.
componentizedPackage	This subpackage type contains one set of files, which form a standardized master or distribution package, e.g. a DCP, an IMF or a DPP.
extraPackage	This subpackage type contains one set of files, which do not belong directly to the audiovisual part of the cinematographic work, but is created to support such work, like press kits, posters etc. Proprietary metadata records from archives databases will also fit here.

`<suuid>` is the unique identifier number (UUID) associated with the subpackage. Each time data are modified inside the subpackage, a new UUID will be created. This ensures a unique identification of data sets.

The subpackage folder contains two mandatory subfolder, `data` and `metadata`, and one optional subfolder `ancillaryData`.

A.3.4 Elements in the subpackage

Movie data elements or extra data elements, which can be images, sound files, a complete movie or something else, dependent on the type, are located in the folder `data`.

For the data elements in the `data` folder – either a single file or a set of files – one technical metadata file has to be created. This file is stored in the `metadata` folder and named `techMD_<suuid>_<no>.xml`. `<suuid>` in the file name is identical with the UUID in the subpackage name. `<no>` is a consecutive number to allow multiple metadata files, starting with index 0 and uses 4 digits. At the moment only one metadata file `techMD_<suuid>_0000.xml` is expected, but for future extensions the number is included in the same way as it is in other file names. `techMD_<suuid>_<no>.xml` files are constructed based on the EBUCore schema with CPP specific extensions.

The data elements are optionally linked to digital provenance data. This information about events and processes of data objects are stored in files called `digiprovMD_<suuid>_<no>.xml` inside the metadata folder.

The most important additional file in the subpackage is the packing list. The packing list `packingList.xml` is the inventory list of the `data` and `ancillaryData` folder and links the data elements to the metadata files in the `metadata` folder. Figure A.3 shows the linking between the files in a subpackage.

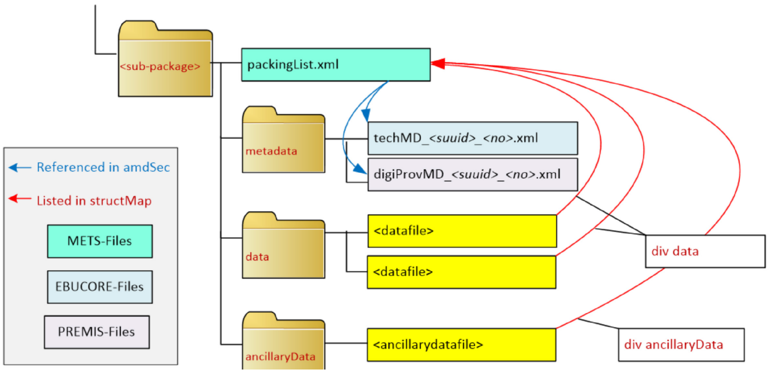

Figure A.3 — Linking inside the subpackage

A.3.5 XML elements and attributes in packing list of subpackages

The packing list `packingList.xml` is constructed based on the METS schema and contains at least the following sections as described in Figure A.4.

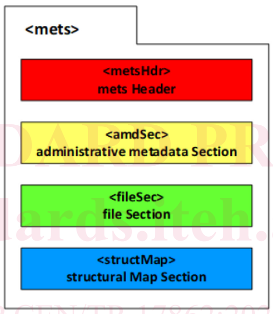

Figure A.4 — METS sections in packingList.xml files

Table A.2 describes the most important elements. A detailed description of the elements and attributes can be found in the METS documentation [3].

Table A.2 — METS elements in packingList.xml files

Element	Type	May contain	Has attributes	Contained within	Min/max
<agent>		<name> <note>	ID ROLE OTHERROLE TYPE OTHERTYPE	<metsHdr>	0/∞
<altRecordID>			ID TYPE	<metsHdr>	0/∞
<amdSec>	amdSecType	<techMD> <digiprovMD>	ID	<mets>	0/∞