E-ARK Archival Information Package (AIP)

Specification for Archival Information Packages

DILCIS BOARD

Date: 15.10.2021

Version: 2.1.0

Executive Summary
This E-ARK AIP format specification defines the requirements for building Archival Information Packages (AIPs) containing the information to be stored by an archive for the long term. While the AIP format inherits general properties from the "Common Specification for Information Packages" (CSIP), the difference to the Submission Information Package (SIP) and Dissemination Information Package (DIP) is the time dimension: The AIP format defines a generic structure for storing both, a series of information packages, i.e. Submission Information Packages (SIPs), which were transferred at different, subsequent points in time, as well as any changes that needed to be applied for preservation reasons. The AIP format can therefore be seen as a wrapper for information packages which allows recording the provenance of the archival entity concerning changes and sequential submissions (SIPs) over time. Furthermore, an important requirement for creating manageable physical AIP packages is to limit the size of the AIPs. Therefore, the AIP specification defines a practice for splitting very large AIPs into multiple, sub-ordinated parts. Finally, the AIP specification gives a best practice recommendations regarding the physical packaging of AIPs.

Preface

I. Aim of the Specification

This document is one of several related specifications which aim to provide a common set of usage descriptions of international standards for packaging digital information for archiving purposes. These specifications are based on common, international standards for transmitting, describing and preserving digital data. They also utilise the Reference Model for an Open Archival Information System (OAIS), which has Information Packages as its foundation. Familiarity with the core functional entities of OAIS is a prerequisite for understanding the specifications.

The specifications are designed to help data creators, software developers, and digital archives to tackle the challenge of short-, medium- and long-term data management and reuse in a sustainable, authentic, cost-efficient, manageable and interoperable way. A visualisation of the current specification network can be seen here:

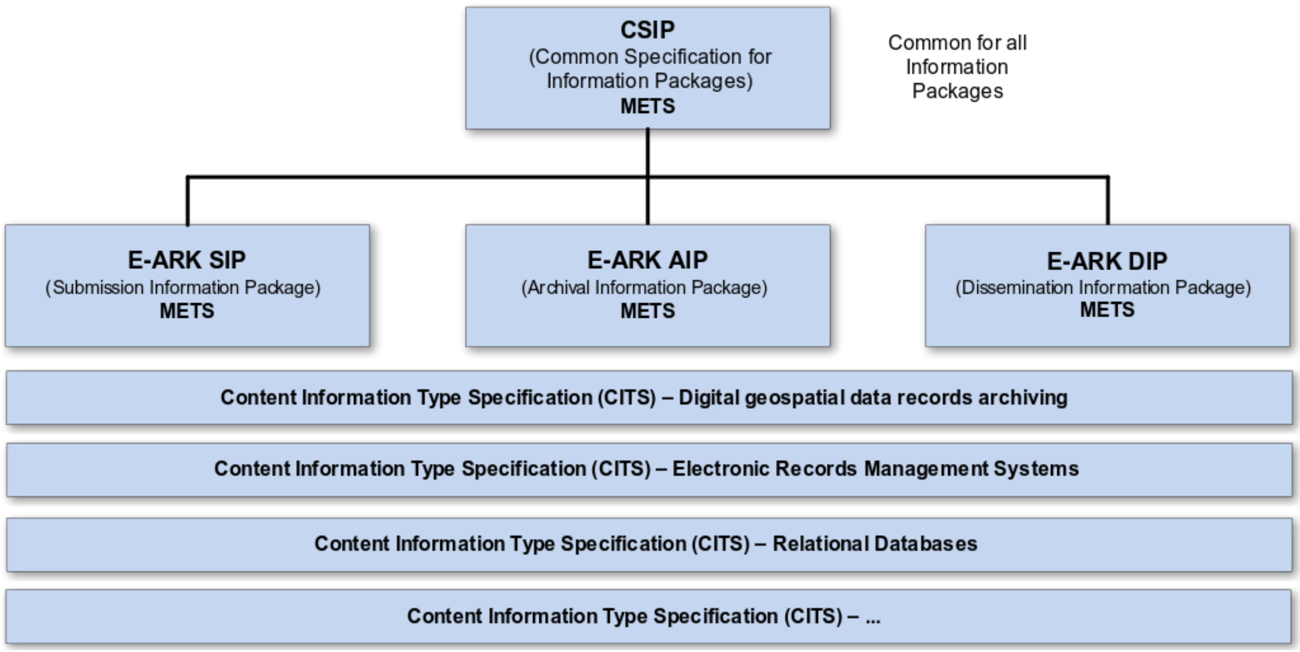
CSIP
(Common Specification for Information Packages)
METS
Common for all Information Packages
E-ARK SIP
(Submission Information Package)
METS
E-ARK AIP
(Archival Information Package)
METS
E-ARK DIP
(Dissemination Information Package)
METS
Content Information Type Specification (CITS) – Digital geospatial data records archiving
Content Information Type Specification (CITS) – Electronic Records Management Systems
Content Information Type Specification (CITS) – Relational Databases
Content Information Type Specification (CITS) – ...

Figure I: Diagram showing E-ARK specification dependency hierarchy. Note that the image only shows a selection of the published CITS and isn't an exhaustive list.

Overview of the E-ARK Specifications

Common Specification for Information Packages (E-ARK CSIP)

This document introduces the concept of a Common Specification for Information Packages (CSIP). The main purposes of CSIP are to:

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

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

Specification for Submission Information Packages (E-ARK SIP)

The main aims of this specification are to:

- Define a general structure for a Submission Information Package format suitable for a wide variety of archival scenarios, such as document and image collections, databases or geospatial data.
- Enhance interoperability between Producers and Archives.
- Recommend best practices regarding the structure, content and metadata of Submission Information Packages.

Specification for Archival Information Packages (E-ARK AIP)

The main aims of this specification are to:

- Define a generic structure of the AIP format suitable for a wide variety of data types, such as document and image collections, archival records, databases or geospatial data.
- Recommend a set of metadata related to the structural and the preservation aspects of the AIP as implemented by the eArchiving Reference Implementation (earkweb).
- Ensure the format is suitable to store large quantities of data.

Specification for Dissemination Information Packages (E-ARK DIP)

The main aims of this specification are to:

- Define a generic structure of the DIP format suitable for a wide variety of archival records, such as document and image collections, databases or geographical data.
- Recommend a set of metadata related to the structural and access aspects of the DIP.

Content Information Type Specifications (E-ARK CITS)

The main aim of a Content Information Type Specification (CITS) is to:

- Define, in technical terms, how data and metadata must be formatted and placed within a CSIP Information Package to achieve interoperability in exchanging specific Content Information.

The number of possible Content Information Type Specifications is unlimited. For a list of existing Content Information Type Specifications see the DILCIS Board webpage (DILCIS Board, http://dilcis.eu/).

II. Organisational Support

This specification is maintained by the Digital Information LifeCycle Interoperability Standards Board (DILCIS Board, http://dilcis.eu/). The role of the DILCIS Board is to enhance and maintain the draft specifications developed in the European Archival Records and Knowledge Preservation Project (E-ARK project, http://eark-project.com/), which concluded in January 2017. The Board consists of eight members, but no restriction is placed on the number of participants taking part in the work. All Board documents and specifications are stored in GitHub (https://github.com/DILCISBoard/), while published versions are made available on the Board webpage. The DILCIS Board have been responsible for providing the core specifications to the Connecting Europe Facility eArchiving Building Block https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eArchiving/.

III. Authors & Revision History

A full list of contributors to this specification, as well as the revision history, can be found in the Postface material.

Table of contents

1 Scope and purpose

To briefly recall the three types of information packages as defined by OAIS (OAIS 2012), there is the Submission Information Package (SIP) which is used to submit digital objects to a repository system; the Archival Information Package (AIP) which allows the transmission of a package to the repository, and its storage over the long-term; and the Dissemination Information Package (DIP) which is used to disseminate digital objects to the requesting user.

This document is specification of the E-ARK Archival Information Package format (E-ARK AIP, subsequently referred to as AIP). It defines requirements and guidelines for creating AIPs which are adequate to store information packages for the long term. The key objectives of this format are to:

- define the AIP format as an extension of the E-ARK CSIP so that it is suitable for the long-term storage of a wide variety of data types, such as document and image collections, archival records, databases or geographical data.
- recommend specific ways of using metadata standards to improve interoperability with regard to the use of long-term archiving standards.
- specify a form of packaging AIP container files while ensuring that the format is suitable for the storage of large quantities of data.

2 Relation to other documents

This specification document originates from the document "D4.4 Final version of SIP-AIP conversion component (Part A: AIP specification)" (Faria et al. 2017) created in the E-ARK project (European Archival Records and Knowledge Preservation) which ran from 2014 to 2017 and was funded by the European Commission as part of the Seventh Framework Programme for Research.

The common requirements for all types of E-ARK information packages are defined by the "Common Specification for Information Packages (CSIP) (see Bredenberg et al. 2018)".

Further documents which are related to the AIP specification in a general sense are listed in the CSIP (section 1.4 "Relation to other documents").

3 Introduction

The AIP format defines an information package for storing archival content that is going to be transferred to a repository for long-term preservation purposes. The AIP format allows keeping a record of changes that are applied to an AIP in form of metadata edits, digital preservation measures (e.g. migration or adding emulation information), or submission updates.[1]

The purpose of defining a standard format for the archival information package is to pave the way for simplified repository migration. Given the increasing amount of digital content archives need to safeguard nowadays, changing the repository solution should be based on a standard exchange format. This is to say that a data repository solution provider does not necessarily have to implement this format as the internal storage format, but it should at least allow exporting AIPs. By this way, the costly procedure of exporting data, producing SIPs, and ingesting them again in the new repository can be simplified. Data repository solution providers know what kind of existing data they can expect if they were chosen to replace an existing repository solution. An E-ARK compliant repository solution should be able to immediately analyse and incorporate existing data in form of AIPs without the need of applying data transformation or having to fulfil varying SIP creation requirements.

Generally, a great variety of repository systems are being developed by different providers, and the way how the AIP is stored depends on specific requirements which have been addressed according to the needs of their respective customers. For this reason, the purpose of this AIP format is not to impose a common storage format that all repository systems need to implement. While it can be used as an archival storage format, it can also be seen as a format that makes system migration easier.

4 Definitions and remarks

4.1 Logical and physical AIP

Definition: The *logical AIP* is the set of digital objects and metadata representing an entire intellectual entity regardless of the physical manifestation.

Definition: The *physical AIP* is the manifestation of a logical AIP in form of one or several container files.

4.2 Version of an AIP

Information packages are permanent: more precisely the information they contain is assumed to be permanent and always describes the same unaltered conceptual entity. Nevertheless, the way in which this information is represented may change.

For the purposes of the AIP format specification, the concept *AIP version* is used as defined by OAIS:

> “AIP Version: An AIP whose Content Information or Preservation Description Information has undergone a Transformation on a source AIP and is a candidate to replace the source AIP. An AIP version is considered to be the result of a Digital Migration. (OAIS 2012, 1–9)”

[1] A *submission update* is a re-submission of an SIP at a later point in time related to an AIP which contains a previous version of this SIP. Section 5.2.1 explains this concept more in detail.

A new version of an AIP can contain one or more new representations which can be either the result of a digital migration or information that enables the creation of an emulation environment to render a representation. Or representation could be removed from the AIP. In both cases the result is the creation of a new version of the AIP.

If the logical AIP is changed, the physical representation of the information in a container may change as well. The result is a new version of the physical container files.

4.3 Segmentation of the AIP

From the point of view of preserving the integrity of the AIP, the ideal case is that the logical AIP is packaged as one single physical container, because all of the metadata and content required to interprete the information package is available in a single entity. In reality, however, this is not always possible because the size of the physical container can become very large.

For this reason, the AIP format describes how to partition the AIP and keep representations or representation parts in separate physical container files (see section 5.1). Admittedly, this puts the integrity of the AIP at risk because in case of disaster recovery the physical container does not represent the entire intellectual entity. Further, dependencies to another (lost) physical container could make it impossible to interpret, understand, or render the content. However, it is a necessary measure if the amount of data exceeds the capacity limitation of long-term storage media.

Definition: Segmentation is a physical manifestation of a logical AIP where a set of physical container files contains parts of the logical AIP. Each segment of the logical AIP is a packaged as a TAR or ZIP file and contains its own structural metadata.

It is mandatory to document the segmentation of an AIP in the structural metadata.

In (OAIS 2012) p. 1-9, the Archival Information Collection (AIC) is described as
“an Archival Information Package whose Content Information is an aggregation of other Archival Information Packages.” The AIC can therefore represent a the structure of a segmented AIP is defined by a header information package (AIC) pointing to the child information packages (AIPs).

It is recommended that the structural metadata of the child information packages record to which header information package (AIC) they belong.

If during the life-cycle of the AIP preservation actions are applied to specific parts, i.e. child packages of the logical AIP, the AIC (header information package) must update the references to the child packages. However, it is not required to update the reference to a parent of a child package which is not concerned by a preservation action.

4.4 Splitting

Definition: Splitting is a special case of segmentation where large files (e.g. large representation content files) are divided into parts of a fixed byte length. However, the splitted content files are wrapped by AIP segments, i.e. they are contained in an AIP which references the parent information package (AIC) to which they belong.

4.5 Differential AIP

A differential package is an incomplete form of the AIP which contains only part of the original AIP it is derived from. The purpose of the differential AIP is to allow persisting updates to a previously stored AIP.

The differential AIP is mostly relevant for the physical container files storing the actual content of the AIP. In case of large AIPs, this allows adding or overriding data or metadata to an physical container containing parts of an AIP or the entire AIP content.

5 AIP format

The AIP format consists of a set of recommendations and requirements[ˆ2] regarding the use of structural and preservation metadata which are introduced in the following.

5.1 AIP specific structural metadata

5.1.1 Compound vs. divided package structure

The ability to manage representations or representation parts separately is required because the digital data submissions can be very large. This is not only relevant for storing the AIP, it also concerns the SIP which might need to be divided before the data is submitted to the repository. In addition, it is important to find and identify AIP segments when creating a DIP which relies on metadata or content of these segments.

In the following, two approaches for defining the structure of the IP will be described with a focus on requirements of the AIP format: the *compound* structure is represented by one single structural metadata file, and the *divided* structure has one structural metadata file that references those of individual representations. An example will help to describe the two alternatives.

If the *compound* METS structure is used, as shown in Figure 3, a single METS file contains all references to metadata and data files contained in the IP.

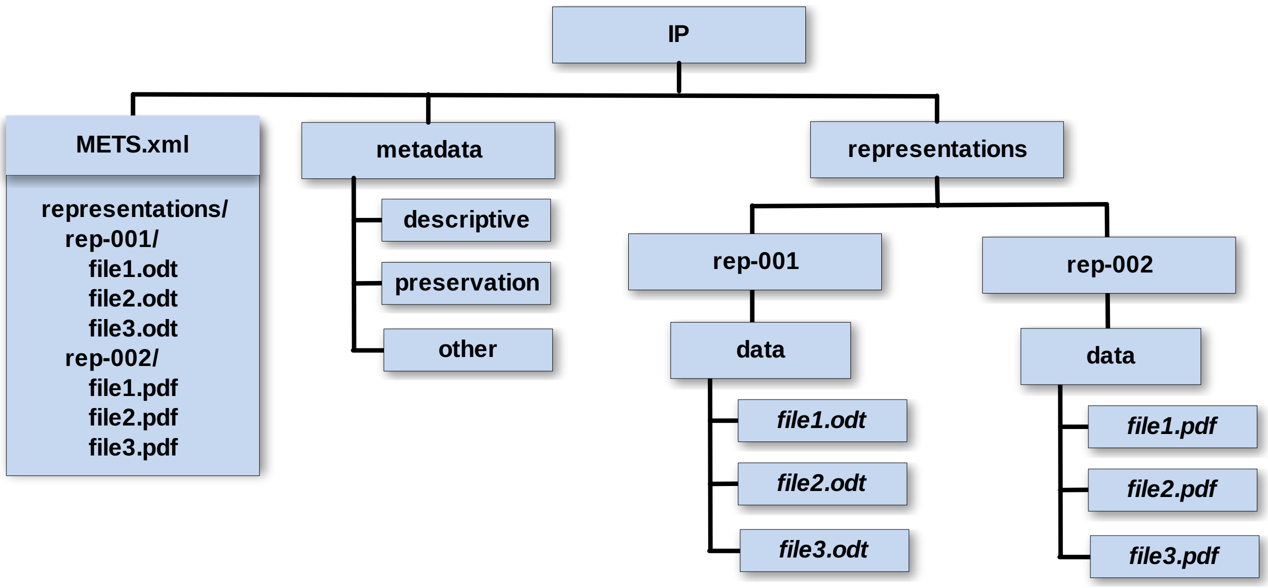

Figure 3: One METS file in the root of the package references all metadata and data files

Even though the number suffix of the folders `rep-001` and `rep-002` of the example shown in Figure 3 suggests an order of representations, there are no requirements regarding the naming of folders containing the representations. The order of representations and the relations between them is defined by the structural and preservation metadata.

If the *divided* METS structure is used, as shown in Figure 4, then a separate METS file for each representation exists which are referenced by the root METS file. The example shown in Figure 4 has a METS file in the IP's root which points to the METS files `Representations/Rep-001/METS.xml` and `Representations/Rep-002/METS.xml`.

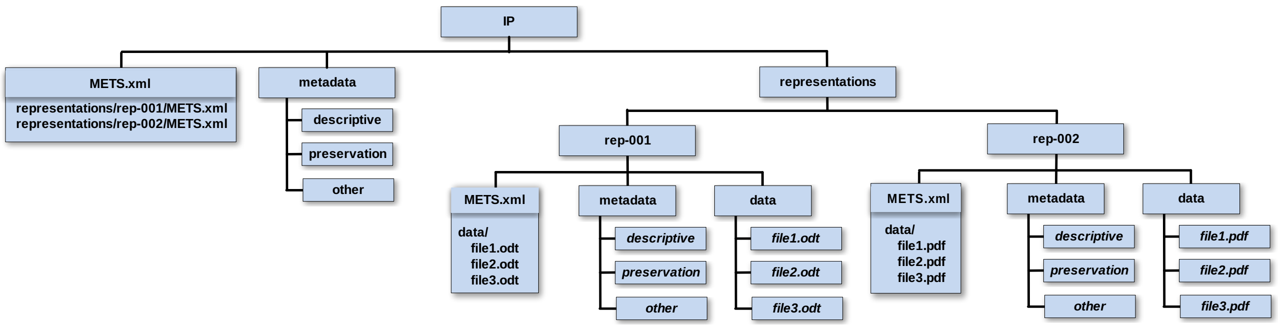

Figure 4: Root METS file references METS files of the different representations

The reason why this alternative was introduced is that it makes it easier to manage representations independently from each other. This can be desired for very large representations, in terms of file size or the amount of files (making the root METS difficult to work with).

As a corollary of this division method we define, a representation-based division as the separation of representations in different folders under the `representations` folder as shown in the example of Figure 4. We also define a size-based division as the separation of representation parts. To illustrate this, Figure 5 shows an example where a set of files belongs to the same representation (here named `binary`) and is referenced in two separate physical containers (here named {C1} and {C2} respectively). A key requirement when using size-based division of a representation is that there must not be any overlap in the structure of the representations, and that each sub-folder path must be unique across the containers where the representation parts together constitute a representation entity. Note that for this reason a numerical suffix is added to the representation METS files, to avoid overwriting representation METS files when automatically merging the divided representation back into one single physical representation.

AIP1: If a representation is divided into parts, the representation component MUST use the same name in the different containers.

AIP2: If a representation is divided into parts, each the sub-paths of items (folders and files) MUST be unique across the different containers. This allows aggregating representation parts without accidentally overwriting folders or files.

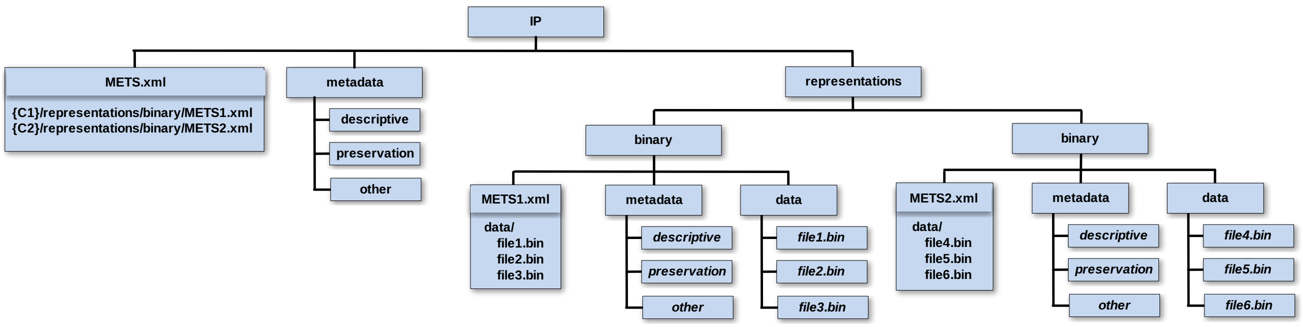

Figure 5: Example of an IP.

For example, let us assume an AIP with two representations, each of which consists of a set of three files. In the first representation all data files are in the Open Document Format (ODT) and in the second one - as a derivative of the first representation - all files are in the Portable Document Format (PDF).

aipstruct

5.1.2 Parent-Child relationship

As already pointed out, the divided METS structure was introduced to support the physical separation of representations or representation parts and allow distributing these components over a sequence of AIPs.

As shown in Figure 6 The composition of a logical AIP can be expressed by a parent-child relationship between AIPs. It is a bidirectional relationship where each child-AIP bears the information about the parent-AIP to which they belong and, vice versa, the parent-AIP references the child-AIPs.

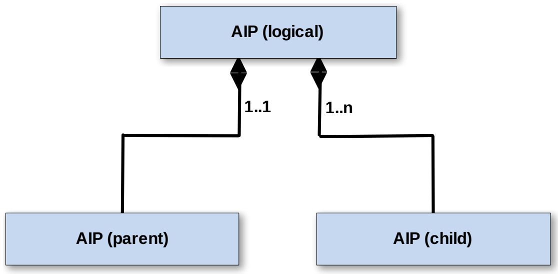

Figure 6: Parent-child relationship between AIPs

Even though this parent-child relationship could be used to create a hierarchical graph of AIPs, the scope of this specification is limited to a flat list of AIPs which are subordinated to one parent-AIP.

Assuming that a new AIP (e.g. containing an additional representation) needs to be added after parent- and child-AIPs have been stored, the recreation of the whole logical AIP might be inefficient, especially if the AIPs are very large. For this reason, existing child-AIPs remain unchanged in case a new version of the parent-AIP is created. Only the new version of the parent-AIP has references to all child-AIPs as illustrated in Figure 7. As a consequence, in order to find all siblings of a single child-AIP it is necessary to get the latest version of the parent-AIP which implies the risk that the integrity of the logical AIP is in danger if the latest version of the parent-AIP is lost.

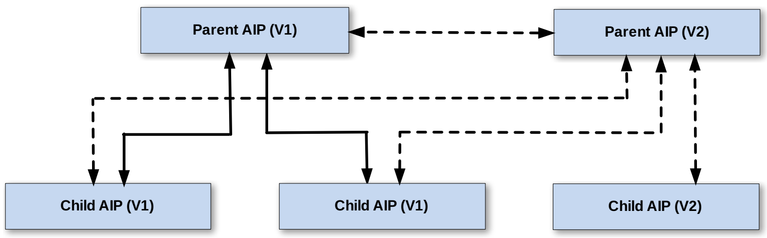

Figure 7: New version of a parent-AIP

The result of this process is a sequence of physical containers of child-AIPs plus one additional parent-AIP. The relation of the AIPs is expressed by means of structural metadata in the METS files.

5.1.3 METS identifier

Each AIP root METS document must be assigned a persistent and unique identifier. Possible identifier schemes are amongst others: OCLC Purls[2], CNRI Handles[3], DOI[4]. Alternatively, it is possible to use a UUID as a locally unique identifier.[5]

Using this identifier, the system must be able to retrieve the corresponding package from the repository.

[2]http://purl.org/docs/index.html
[3]http://www.handle.net
[4]https://www.doi.org
[5]Universally Unique Identifier according to RFC 4122, http://tools.ietf.org/html/rfc4122.html

According to the Common Specification, any ID element must start with a prefix (also, the XML ID data type does not permit IDs that start with a number, so a prefix solves this issue).

We recommend using an internationally recognized standard identifier for the institution from which the SIP originates as a prefix. This may lead to problems with smaller institutions, which do not have any such internationally recognized standard identifier. We propose in that case, to start the prefix with the internationally recognized standard identifier of the institution, where the AIP is created, augmented by an identifier for the institution from which the SIP originates.

An alternative to this is to use a UUID:

```
https://tools.ietf.org/html/rfc4122
```

The prefix `urn:uuid:` would indicate the identifier type. For example, if the package identifier value is `"123e4567-e89b-12d3-a456-426655440000"` this would be the value of the METS root element's `OBJID` attribute:

```
/mets/@OBJID="urn:uuid:123e4567-e89b-12d3-a456-426655440000"
```

The `OBJID` attribute of the root METS is the persistent unique identifier of the AIP.

Structural map of a divided METS structure

AIP3: When an AIP uses the divided METS structure, i.e. the different representations have their own METS file, the mandatory `<structMap>` MUST organize those METS files through `<mptr>` and `<fptr>` entries, for each representation. The `<mptr>` node MUST reference the `/<representation>/METS.xml` and point at the corresponding `<file>` entry in the `<fileSec>` using the `<fptr>` element.

```
<structMap ID="uuid-1465D250-0A24-4714-9555-5C1211722FB8" TYPE="PHYSICAL" LABEL=
    "CSIP structMap">
   <div ID="uuid-638362BC-65D9-4DA7-9457-5156B3965A18" LABEL="uuid-4422c185
      -5407-4918-83b1-7abfa77de182">
     <div LABEL="representations/images_mig-1">
       <mptr xlink:href="./representations/images_mig-1/METS.xml" xlink:title="
          Mets file describing representation: images_mig-1 of AIP: urn:uuid:
          d7ef386d-275b-4a5d-9abf-48de9c390339." LOCTYPE="URL" ID="uuid-c063ebaf
          -e594-4996-9e2d-37bf91009155"/>
       <fptr FILEID="uuid-fb9c37e7-1c90-4849-a052-1875e67853d5"/>
     </div>
     <div LABEL="representations/docs_mig-1">
       <mptr xlink:href="./representations/docs_mig-1/METS.xml" xlink:title="
          Mets file describing representation: docs_mig-1 of AIP: urn:uuid:
          d7ef386d-275b-4a5d-9abf-48de9c390339." LOCTYPE="URL" ID="uuid-335f9e55
          -17b2-4cff-b62f-03fd6df4adbf"/>
       <fptr FILEID="uuid-3f2268cd-7da9-4ad8-909b-4f17730dacaf"/>
     </div>
   </div>
</structMap>
```

Listing 1: Structural map referencing METS files of the different representations

5.1.4 Metadata representation of the AIP structure

Child AIP references parent AIP

The optional reference to a parent AIP is expressed by a structural map with the LABEL attribute value `Parent`. Listing 2 shows an example where a UUID is used as the package identifier and the `xlink:href` attribute has the UUID identifier value of the referenced parent AIP as value. This identifier implicitly references the METS file of the corresponding package. If other locator types, such as URN, URL, PURL, HANDLE, or DOI are used, the `LOCTYPE` attribute can be set correspondingly.

```
<structMap ID="uuid-35CB3341-D731-4AC3-9622-DB8901CD6736" TYPE="PHYSICAL" LABEL=
    "parent AIP">
    <div ID="uuid-35CB3341-D731-4AC3-9622-DB8901CD6738" LABEL="AIP parent
        identifier">
      <mptr xlink:href="urn:uuid:3a487ce5-63cf-4000-9522-7288e208e2bc"
            xlink:title="Referencing the parent AIP of this AIP
                         (URN:UUID:3218729b-c93c-4daa-ad3c-acb92ab59cee)."
            LOCTYPE="OTHER" OTHERLOCTYPE="UUID"
            ID="uuid-755d4d5f-5c5d-4751-9652-fcf839c7c6f2"/>
    </div>
</structMap>
```

Listing 2: Using a structMap to reference the parent AIP

Parent AIP references child AIPs

The parent AIP which is referenced by child AIPs must have a structural map listing all child AIPs. Listing 3 shows the structural map of a parent AIP listing four child AIPs.

```
<structMap TYPE="PHYSICAL" LABEL="child AIPs">
    <div LABEL="child AIPs">
        <div LABEL="child AIP">
            <mptr xlink:href="urn:uuid:cea73348-741d-4594-ab8f-0b9e652c1099"
```

```
                xlink:title="Referencing a child AIP."
                LOCTYPE="OTHER" OTHERLOCTYPE="UUID"
                ID="uuid-d98e416f-55a7-4237-8d45-59c22d221669"/>
        </div>
        <div LABEL="child AIP">
            <mptr xlink:href="urn:uuid:cea73348-741d-4594-ab8f-0b9e652c1099"
                xlink:title="Referencing a child AIP."
                LOCTYPE="OTHER" OTHERLOCTYPE="UUID"
                ID="uuid-70f8ec28-23f1-4364-9163-b3e99165b6e6"/>
        </div>
        <div LABEL="child AIP">
            <mptr xlink:href="urn:uuid:3218729b-c93c-4daa-ad3c-acb92ab59cee"
                xlink:title="Referencing a child AIP."
                LOCTYPE="OTHER" OTHERLOCTYPE="UUID"
                ID="uuid-77373d7f-e241-481b-bf89-675335beb049"/>
        </div>
        <div LABEL="child AIP">
            <mptr xlink:href="urn:uuid:cea73348-741d-4594-ab8f-0b9e652c1099"
                xlink:title="Referencing a child AIP."
                LOCTYPE="OTHER" OTHERLOCTYPE="UUID"
                ID="uuid-3f0cc05c-f27d-499d-a6fd-63bdfed13cb0"/>
        </div>
    </div>
</structMap>
```

Listing 3: Using a structMap to reference the parent AIP

5.2 AIP relevant preservation metadata

As already mentioned, PREMIS (PREMIS 2017) is used to describe technical metadata of digital objects, rights metadata to define the rights status in relation to specific agents or for specific objects, and to record events that are relevant regarding the digital provenance of digital objects.

Regarding general use of PREMIS, there is the E-ARK Content Information Type Specification for Preservation Metadata using PREMIS[6]

In the following, only the PREMIS elements which are relevant for the AIP format are described. **NOTE:** in the listings showing PREMIS code parts, the prefix "premis" is omitted (default namespace is the PREMIS namespace[7]) while the "mets" prefix is explicitly added if a relation to the METS file is explained.

[6]https://citspremis.dilcis.eu/specification/CITS_Preservation_metadata_v1.0.pdf
[7]Namespace: http://www.loc.gov/premis/v3, namespace schema location: http://www.loc.gov/standards/premis/premis.xsd

5.2.1 PREMIS object

The PREMIS object contains technical information about a digital object.

File format

AIP7: The format element COULD be provided either using the formatRegistry or the formatDesignation element sub-elements, or both.

AIP8: Regarding the formatRegistry, the Persistent Unique Identifier (PUID)[8] based on the PRONOM technical registry[9] COULD be used.

An example is shown in Listing 6.

```
<format>
    <formatDesignation>
        <formatName>XML</formatName>
        <formatVersion>1.0</formatVersion>
    </formatDesignation>
    <formatRegistry>
        <formatRegistryName>PRONOM</formatRegistryName>
        <formatRegistryKey>fmt/101</formatRegistryKey>
        <formatRegistryRole>specification</formatRegistryRole>
    </formatRegistry>
</format>
```

Listing 6: Optionally, the format version can be provided using the `formatDesignation` element.

Storage

AIP11: The storage element COULD hold contain information about the physical location of the digital object.

Ideally this is a resolvable URI, but it can also generally hold information needed to retrieve the digital object from the storage system (e.g. access control or for segmented AIPs).

An example is shown in Listing 9.

```
<storage>
    <contentLocation>
        <contentLocationType>URI</contentLocationType>
        <contentLocationValue>
          /path/to/file.txt
```

[8]http://www.nationalarchives.gov.uk/aboutapps/pronom/puid.htm
[9]http://www.nationalarchives.gov.uk/PRONOM

```
        </contentLocationValue>
    </contentLocation>
    <storageMedium>hard disk HD2253</storageMedium>
</storage>
```

Listing 9: Storage description

Relationship

AIP12: The `relationship` element SHOULD be used to describe relationships of the digital object.

AIP13: If an AIP is part of another AIP, then the element `relationshipSubType` MUST reference the superordinate AIP.

An example of the latter case is shown in Listing 10.

```
<relationship>
    <relationshipType>structural</relationshipType>
    <relationshipSubType>is included in</relationshipSubType>
    <relatedObjectIdentification>
            <relatedObjectIdentifierType>repository</relatedObjectIdentifierType
   >
            <relatedObjectIdentifierValue>
               ID123e4567-e89b-12d3-a456-426655440000
        </relatedObjectIdentifierValue>
    </relatedObjectIdentification>
</relationship>
```

Listing 10: Relationship

5.2.2 PREMIS event

Event identifier

AIP15: The `eventIdentifier` SHOULD be used to identify events, such as preservation actions, which were applied. An example is shown in Listing 12.

```
<eventIdentifier>
    <eventIdentifierType>local</eventIdentifierType>
    <eventIdentifierValue>PDF to PDF/A</eventIdentifierValue>
</eventIdentifier>
```

Listing 12: Event identifier

Link to agent/object

AIP16: If an event is described, the agent which caused the event (e.g. person, software, hardware, etc.) MUST be related to the event by means of the `linkingAgentIdentifier` element.

In the example shown in listing 20 the SIP to AIP conversion software is linked as agent with identifier value 'Sip2Aip' and the corresponding object is linked by the local UUID value. An example is shown in Listing 13.

```
<linkingAgentIdentifier>
    <linkingAgentIdentifierType>local</linkingAgentIdentifierType>
        <linkingAgentIdentifierValue>
            IngestSoftware
        </linkingAgentIdentifierValue>
    </linkingAgentIdentifier>
    <linkingObjectIdentifier>
    <linkingObjectIdentifierType>local</linkingObjectIdentifierType>
        <linkingObjectIdentifierValue>
            metadata/file.xml
        </linkingObjectIdentifierValue>
</linkingObjectIdentifier>
```

Listing 13: Link to agent/object

Migration event type

AIP17: The event by which a resource was created SHOULD to be recorded by means of the `relatedEventIdentificatio` element.

An example is shown in Listing 14.

```
<event>
    <eventIdentifier>
        <eventIdentifierType>local</eventIdentifierType>
        <eventIdentifierValue>migration-001</eventIdentifierValue>
    </eventIdentifier>
    <eventType>MIGRATION</eventType>
    <eventDateTime>2015-09-01T01:00:00+01:00</eventDateTime>
    <eventOutcomeInformation>
        <eventOutcome>success</eventOutcome>
    </eventOutcomeInformation>
    <linkingAgentIdentifier>
        <linkingAgentIdentifierType>local</linkingAgentIdentifierType>
        <linkingAgentIdentifierValue>
            FileFormatConversion001
        </linkingAgentIdentifierValue>
    </linkingAgentIdentifier>
    <linkingObjectIdentifier>
        <linkingObjectIdentifierType>local</linkingObjectIdentifierType>
        <linkingObjectIdentifierValue>
            metadata/file.xml
        </linkingObjectIdentifierValue>
    </linkingObjectIdentifier>
    <relatedEventIdentification>
        <relatedEventIdentifierType>local</relatedEventIdentifierType>
        <relatedEventIdentifierValue>
            ingest-001
        </relatedEventIdentifierValue>
    </relatedEventIdentification>
</event>
```

Listing 14: Migration event

The event shown in Listing 15 expresses the fact that the object `metadata/file.xml` is the result of the migration event "migration-001" and the event which created the source object is "ingest-001".

5.2.3 PREMIS agent

AIP18: Agents which are referenced in events must be described by means of the `agent` element.

Listing 15 shows a software for indexing named `IndexingSoftware` which supports full text search of the items contained in a package. In this case, the "discovery right" is assigned to this agent.

```
<agent>
    <agentIdentifier>
        <agentIdentifierType>local</agentIdentifierType>
        <agentIdentifierValue>Indexer</agentIdentifierValue>
    </agentIdentifier>
    <agentName>IndexingSoftware</agentName>
    <agentType>Software</agentType>
    <linkingRightsStatementIdentifier>
        <linkingRightsStatementIdentifierType>
            local
        </linkingRightsStatementIdentifierType>
        <linkingRightsStatementIdentifierValue>
            discovery-right-001
        </linkingRightsStatementIdentifierValue>
    </linkingRightsStatementIdentifier>
```

```
</agent>
```

Listing 15: Software as an agent

5.3 Physical Container Packaging

This part of the AIP format specification gives recommendations regarding the creation of the physical packaging of the logical AIP into either one or multiple transferable and storable entities.

5.3.1 Naming scheme for physical containers

The recommended naming of phyiscal containers differentiates 4 levels which can be reflected in the file name:

- Identifier part (character-mapped unique identifier)
- Version label (based on version number)
- Bag label (based on Bag number)
- Differential label (differential Package)

The identifier is a name that uniquely identifies the AIP. Any physical container file which belongs to the same AIP should start with a file name part that is based on this identifier.

The "version label" denotes the version of an AIP and could be added as a suffix to the identifier part.

The "bag label" represents a part of the AIP. This is optional in case the AIP is divided into several parts, e.g. because a policy prescribes a maximum file size limit for physical container files.

The "differential label" represents a differential package where files and folders of a physical container files are complemented or overridden by files or folders of a differential package.

In the following the concepts of the naming scheme will be introduced. Concrete examples of how the naming scheme can be applied in specific digital preservation scenarios related to the life-cycle of an AIP are listed in Appendix E.

Identifier part

According to the requirement defined in section 5.3.1 ("METS identifier"), every AIP bears an identifier which must be recorded in the root METS file of the AIP. By definition, this identifier is the identifier of the AIP itself.

AIP20:: The identifier of the AIP – defined by the attribute `OBJID` of the root METS file's root element SHOULD be used to derive the beginning part of the file name of the physical storage container.

The file name part which is derived from the AIP's identifier is called the *AIP file name ID*.

AIP21: A specified policy SHOULD be defined which allows deriving a cross-platform, portable file name part from the AIP's identifier and, vice versa, to infer the identifier from the physical container's filename.

A first option to implement this requirement would be to limit the characters used in the file name to the "Portable Filename Character Set"[10] which only allows the following character set for saving files:

- Uppercase A to Z
- Lowercase a to z
- Numbers 0 to 9
- Period (.)
- Underscore (_)
- Hyphen (-)

If the identifier of the AIP had characters which do not fall into this character set, then these would need to be mapped into specific ones of the accepted character set.

One proposed way to achieve a bi-directional mapping between identifiers and file names is the pairtree character mapping specification.[11]

AIP22: The file name of the physical container file SHOULD start with a unique name of the AIP which is equal for to all versions and parts that belong to the same logical AIP.

For example, let us assume the identifier of the AIP was:

```
"urn:uuid:123e4567-e89b-12d3-a456-426655440000"
```

Then this identifier string would be converted to the folder name because ": -> +" is defined as a single-character to single-character conversion:

```
"urn+uuid+123e4567-e89b-12d3-a456-426655440000"
```

The packaged entity should also bear this name, e.g. packaged using TAR the name would be:

```
"urn+uuid+123e4567-e89b-12d3-a456-426655440000.tar"
```

In this example, the AIP's physical container file name only consists of the AIP file name ID.

Version label

AIP23: A suffix COULD be added to the physical container file that bears information about the version of the physical container file. This suffix starts with the character "v" followed by a sequential number where higher numbers represent later versions of the AIP.

[10] http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap03.html#tag_03_282
[11] https://tools.ietf.org/html/draft-kunze-pairtree-01 (see section 3: "Identifier string cleaning")

For example, a version number could be added as a suffix to the AIP file name ID as follows:

```
"urn+uuid+123e4567-e89b-12d3-a456-426655440000_v1.tar"
```

AIP24: The first version of a submission could 0. Any changes applied (e.g. due to preservation measures) could be stored in subsequent versions.

Bag label

AIP25: If an AIP is divided into different phyiscal container files container, a suffix COULD be added to the name which denotes the corresponding part.

For example, the first part of an AIP could be added as a suffix to the AIP file name ID as follows:

```
"urn+uuid+123e4567-e89b-12d3-a456-426655440000_v1_b1.tar"
```

where the character `b` stands for the first bag which contains the part of the AIP (more on the use of bags and the bagit packaging format will be described in the next section).

Differential label

AIP26: A "differential" package contains files and folders which complement or override parts of a complete physical container file. This container file represents an intermediate state of the information packages which should be consolidated, i.e. the differential package should be merged with the physical container file which contains the last complete state of the physical container.

Note that this is valid for adding or updating content, but it is not possible to define a differential package which removes content from the physical container file it refers to.

For example, the differential part of a physical container file could be added as a suffix to the AIP physical container file name as follows:

```
"urn+uuid+123e4567-e89b-12d3-a456-426655440000_v1_b1_d1.tar"
```

where the character `d` stands for the first differential physical container file which relates to the first bag of version 1 of the AIP.

5.3.2 Packaging

Recommended formats for packaging AIPs are TAR and ZIP which are both widely used archive formats.

For both formats there are software utilities that can be used to bundle up files into one file for being able to transfer archival packages.

AIP27: The package content MUST be contained in a single folder.

This means that if the packaged AIP is unpackaged, the content MUST be extracted into a single folder which contains the individuals files and folders.

As an example, let's assume a TAR file with the following name:

```
"urn+uuid+123e4567-e89b-12d3-a456-426655440000_v0_b1.tar"
```

If it is extracted, a folder `urn+uuid+123e4567-e89b-12d3-a456-426655440000_v0_b1` could with the actual AIP content is created.

AIP28: If TAR is used as the packaging format, the content SHOULD be aggregated without using compression.

For example, to create a TAR archive without compression for the AIP folder `"urn+uuid+123e4567-e89b-12d3-a456-426655440000"` using the `tar` utility:

```
tar -cf "urn+uuid+123e4567-e89b-12d3-a456-426655440000_v0_b1.tar" "urn+uuid+123
    e4567-e89b-12d3-a456-426655440000_v0_b1"
```

BagIt

The BagIt[12] format specifies a set of hierarchical file layout conventions for storage and transfer of arbitrary digital content. It can be used for packaging the AIP.

AIP29: As defined by the BagIt specification, the `bagit.txt` file in the root folder MUST contain the BagIt version and tag file character encoding.

```
BagIt-Version: 0.97
Tag-File-Character-Encoding: UTF-8
```

AIP30: A `bagit-info.txt` MUST be available and valid according to the E-ARK BagIt profile (corresponding to the version of this specification).[13]

Example of a `bagit-info` file:

```
BagIt-Profile: https://github.com/DILCISBoard/E-ARK-AIP/blob/{version-tag}/
    profiles/bagit/e-ark-bag-profile.json
Source-Organization: Example Organisation
Organization-Address: {Street}, {City}, {Country}
External-Identifier: urn:uuid:123e4567-e89b-12d3-a456-426655440000
External-Description: Example container.
```

[12]https://tools.ietf.org/html/rfc8493
[13]https://github.com/DILCISBoard/E-ARK-AIP/blob/{version-tag}/profiles/bagit/e-ark-bag-profile.json

```
Bagging-Date: 2018-12-18
Bag-Size: 2.7 MB
Payload-Oxum: 2791644.35
E-ARK-Package-Type: AIP
E-ARK-Specification-Version: 1.1
```

AIP31: The containing folder of the AIP SHOULD be located in the `data` folder as shown Listing 17.

```
urn+uuid+123e4567-e89b-12d3-a456-426655440000/
|- bagit.txt
|- data
|   |- urn+uuid+123e4567-e89b-12d3-a456-426655440000
|       |- metadata
|       |   |- preservation
|       |       |- premix.xml
|       |- METS.xml
|       |- representations
|           |- rep1
|               |- data
|               |   |- Example1.docx
|               |- METS.xml
|- manifest-md5.txt
```

Listing 17: AIP in the `data` folder of a BagIt container.

AIP32: If the AIP is a parent-AIP with the content of the logical AIP distributed over multiple child-AIPs, then the `fetch.txt` file[14] SHOULD contain a list of URLs referencing the child-AIP packages.

AIP33: If a `fetch.txt` file with a list of child-AIPs is used, then every child-AIP file listed in the fetch file SHOULD be listed in every payload manifest.

OCFL

The Oxford Common File Layout (OCFL) specification[15] allows describing the storage structure of an AIP's physical container files.

It is an optional extension which can be used in addition to the packaging and file naming recommendations.

The purpose of the OCFL recommendation is to:

- define standards and conventions for storing and exporting versioned AIPs (AIP life-cycle).

[14]https://tools.ietf.org/id/draft-kunze-bagit-08.html#rfc.section.2.2.3
[15]https://ocfl.io

- enable storing or exporting large amounts of archival content in form of AIP container files to file system storage
- support advanced use cases, such as splitting large information packages and differential AIPs (including removal of content using differential packages).

Listing 18 gives an example of an AIP (version 0) using OCFL. It is based on the OCFL Draft 2021[16] and the BagIt standard file system layout for storage and transfer as defined by RFC8493[17].

```
urn+uuid+1017cc9b-eaed-4064-947e-a07c752d3760
|- 0=ocfl_object_1.0
|- inventory.json
|- inventory.json.sha512
|- v0
    |- content
    |- urn+uuid+1017cc9b-eaed-4064-947e-a07c752d3760_v0_b00001
      |- bag-info.txt
      |- bagit.txt
      |- data
      |    |- metadata
      |    |    |- descriptive
      |    |    |    |- ead.xml
      |    |    |    |- metadata.json
      |    |    |- preservation
      |    |         |- premis.xml
      |    |- METS.xml
      |    |- representations
      |         |- 9799fdd1-57b5-48e3-ba53-2705cc874a00
      |         |- data
      |         |    |- example.pdf
      |         |- metadata
      |         |    |- preservation
      |         |         |- premis.xml
      |         |- METS.xml
      |- manifest-sha256.txt
      |- manifest-sha512.txt
      |- tagmanifest-sha256.txt
      |- tagmanifest-sha512.txt
```

Listing 18: OCFL file listing of an AIP (unpackaged container file)

Note that the OCFL Object includes all versions – v0, v1, … - of the AIP and that one bagit container or several

[16]https://ocfl.io/draft/spec/
[17]https://datatracker.ietf.org/doc/html/draft-kunze-bagit-17

bagit containers (segmentation!) are managed as one OCFL object (See in OCFL 5.4 BagIt in an OCFL Object[18]). This is especially relevant for non-redundant storing of AIPs (the concept of a "differential AIP") and for package segmentation.

Also note that the exmaple in Listing 19 is the "unpackaged" version where the bagit container itself is not packaged.

The packaged version

```
urn+uuid+1017cc9b-eaed-4064-947e-a07c752d3760
|- 0=ocfl_object_1.0
|- inventory.json
|- inventory.json.sha512
|- v0
    |- content
      |- urn+uuid+1017cc9b-eaed-4064-947e-a07c752d3760_v0_b00001.tar
```

Listing 19: OCFL file listing of an AIP (packaged container file)

Note that serialization has been removed from the BagIt specification after version 14 (from 2017, current version is 17) to narrow the scope of the specification. In BagIt Version 14 Section serialization was still included which defined the following requirements:

- The top-level directory of a serialization MUST contain only one bag.
- The serialization SHOULD have the same name as the bag's base directory, but MUST have an extension added to identify the format.
- A bag MUST NOT be serialized from within its base directory, but from the parent of the base directory.

- The deserialization of a bag MUST produce a single base directory bag.

The content of the OCFL object file `0=ocfl_object_1.0` in the listing is shown in Listing 20.

```
ocfl_object_1.0
```

Listing 20: OCFL file listing of an AIP (packaged container file)

And an example for the content of the `inventory.json` is is shown in Listing 21.

```
{
    "digestAlgorithm": "sha512",
    "fixity": {
        "md5": {
            "e5ad509db4ddb4cef0de4c1c19c7988b": [
```

[18]https://ocfl.io/draft/spec/#example-bagit-in-ocfl

```
                 "00000/content/urn+uuid+1017cc9b-eaed-4064-947e-
                    a07c752d3760_v0_b00001.tar"
             ]
         },
         "sha256": {
             "68a5b60ddef62758389f6894a1e7df28c1d228a5d56d2eec3ce2f74e80c27910":
                 [
                 "00000/content/urn+uuid+1017cc9b-eaed-4064-947e-
                    a07c752d3760_v0_b00001.tar"
             ]
         }
     },
     "head": "v0",
     "id": "urn:uuid:1017cc9b-eaed-4064-947e-a07c752d3760",
     "manifest": {
         "24
            db03a2a7d9c7e2e7ea533e2ac84b7274f937eaff31e95f508cd9c5418a902adf5c18d2f67fa80a
            ": [
             "00000/content/urn+uuid+1017cc9b-eaed-4064-947e-
                a07c752d3760_v0_b00001.tar"
         ]
     },
     "type": "https://ocfl.io/1.0/spec/#inventory",
     "versions": {
         "v0": {
             "created": "2021-03-27T18:49:22Z",
             "message": "Original SIP",
             "state": {
                 "24
                    db03a2a7d9c7e2e7ea533e2ac84b7274f937eaff31e95f508cd9c5418a902adf5c18d2
                    ": [
                     "00000/content/urn+uuid+1017cc9b-eaed-4064-947e-
                        a07c752d3760_v0_b00001.tar"
                 ]
             }
         }
     }
}
```

Listing 21: OCFL file listing of an AIP (packaged container file)

At the time of finalizing this specification, the OCFL standard does not support the listing of packaged container files in the inventory file. This would allow using the inventory to document the actual content of physical container files and may follow in a future version of the AIP specification.

6 Appendices

6.1 Appendix A - METS referencing representation METS files

```
<fileSec>
  <fileGrp USE="Common Specification root" ID="uuid-0d4f09a8-0734-49fb-9bea-
   dbf6a3f5a444">
    <file MIMETYPE="application/xml" USE="Datafile" CHECKSUMTYPE="SHA-256"
        CREATED="2016-12-14T09:15:24" CHECKSUM="8
        d3f057ac0e45ef173f9ecbfc432b994415c405259aff694632925faf108f541" ID="uuid
        -3af3e474-991a-4aad-b453-ed3f91d54280" SIZE="2855">
      <FLocat xlink:href="./representations/images_mig-1/METS.xml" xlink:type="
          simple" LOCTYPE="URL"/>
    </file>
    <file MIMETYPE="application/xml" USE="Datafile" CHECKSUMTYPE="SHA-256"
        CREATED="2016-12-14T09:15:24" CHECKSUM="81
        e028df7468ea611b0714148cb607ec74fe1e7914bd762605f38631d21281e9" ID="uuid-
        e1df6f8b-8cc0-442d-bc45-e61724c63372" SIZE="2873">
      <FLocat xlink:href="./representations/docs_mig-1/METS.xml" xlink:type="
          simple" LOCTYPE="URL"/>
    </file>
  </fileGrp>
</fileSec>
<structMap TYPE="physical" LABEL="CSIP structMap">
  <div LABEL="urn:uuid:7ff70669-73a0-4551-ad5b-12ed9b229e38">
    <div LABEL="submission">
      <!-- removed to improve readability -->
    </div>
    <div LABEL="metadata">
      <!-- removed to improve readability -->
    </div>
    <div LABEL="schemas">
      <!-- removed to improve readability -->
    </div>
    <div LABEL="representations"/>
    <div LABEL="representations/images_mig-1">
      <mptr xlink:href="./representations/images_mig-1/METS.xml" xlink:title="
          Mets file describing representation: images_mig-1 of AIP: urn:uuid:7
          ff70669-73a0-4551-ad5b-12ed9b229e38." LOCTYPE="URL" ID="uuid-0799bb22-
          b3b1-4661-b32d-5c2dae0341f9"/>
      <fptr FILEID="uuid-3af3e474-991a-4aad-b453-ed3f91d54280"/>
    </div>
    <div LABEL="representations/docs_mig-1">
      <mptr xlink:href="./representations/docs_mig-1/METS.xml" xlink:title="Mets
          file describing representation: docs_mig-1 of AIP: urn:uuid:7ff70669
         -73a0-4551-ad5b-12ed9b229e38." LOCTYPE="URL" ID="uuid-cc2c70c5
         -9712-4697-834c-5d5acad47f49"/>
      <fptr FILEID="uuid-e1df6f8b-8cc0-442d-bc45-e61724c63372"/>
    </div>
  </div>
</structMap>
```

6.2 Appendix B – METS describing a representation

```
<mets xmlns:ext="ExtensionMETS" 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.ra.ee/METS/v01/IP.xml" TYPE="AIP" OBJID="urn:
   uuid:docs_mig-1" LABEL="METS file describing the AIP matching the OBJID." xsi
   :schemaLocation="http://www.loc.gov/METS/ ../../schemas/mets_1_11.xsd http://
   www.w3.org/1999/xlink ../../schemas/xlink.xsd">
  <metsHdr RECORDSTATUS="NEW" CREATEDATE="2016-12-14T09:15:24">
    <agent TYPE="OTHER" ROLE="CREATOR" OTHERTYPE="SOFTWARE">
      <name>E-ARK earkweb</name>
      <note>VERSION=0.0.1</note>
    </agent>
    <metsDocumentID>METS.xml</metsDocumentID>
  </metsHdr>
  <amdSec ID="uuid-facb861c-5f25-43f7-a1a4-86dfa345a119">
    <digiprovMD ID="uuid-c4113098-6eb5-43f5-9618-6f33ef442400">
      <mdRef MIMETYPE="application/xml" xlink:href="./metadata/preservation/
         premis.xml" LOCTYPE="URL" CREATED="2016-12-14T09:15:24" CHECKSUM="
         d9e3bdc2c2e1d1a07cd88585dfddad62cdf40ca060e09456efc68bd2dc88e3a9" xlink
         :type="simple" ID="uuid-2c990270-d140-4d92-8bca-629e21926535" MDTYPE="
         PREMIS" CHECKSUMTYPE="SHA-256"/>
    </digiprovMD>
  </amdSec>
  <fileSec>
    <fileGrp USE="Common Specification representation urn:uuid:docs_mig-1" ID="
   uuid-cee0bbc3-ac88-4f21-834e-2c06104141ac">
      <file MIMETYPE="application/pdf" USE="Datafile" CHECKSUMTYPE="SHA-256"
         CREATED="2016-12-14T09:15:05" CHECKSUM="
         d50fe727b6bed7b04569671a46d4d8a56b93c295afb69703b14c0544286ff86c" ID="
```

```
          uuid-cf9818bb-567b-44ee-88d8-60a1420feae3" SIZE="2530049">
        <FLocat xlink:href="./data/Suleiman the Magnificent.pdf" xlink:type="
          simple" LOCTYPE="URL"/>
      </file>
      <file MIMETYPE="application/pdf" USE="Datafile" CHECKSUMTYPE="SHA-256"
          CREATED="2016-12-14T09:15:12" CHECKSUM="3824
          fb493235e94bcca3baf33c93a9e4f62d4af387ce055560f01c274ef63da9" ID="uuid
          -3b0e4dcb-727a-44d1-af24-d35676b02bed" SIZE="7603618">
        <FLocat xlink:href="./data/Charlemagne.pdf" xlink:type="simple" LOCTYPE=
          "URL"/>
      </file>
    </fileGrp>
  </fileSec>
  <structMap TYPE="physical" LABEL="CSIP structMap">
    <div LABEL="docs_mig-1">
      <div LABEL="metadata">
        <fptr FILEID="uuid-2c990270-d140-4d92-8bca-629e21926535"/>
      </div>
      <div LABEL="data">
        <fptr FILEID="uuid-cf9818bb-567b-44ee-88d8-60a1420feae3"/>
        <fptr FILEID="uuid-3b0e4dcb-727a-44d1-af24-d35676b02bed"/>
      </div>
    </div>
  </structMap>
  <structMap TYPE="logical" LABEL="Simple AIP structuring">
    <div LABEL="Package structure">
      <div LABEL="metadata files">
        <fptr FILEID="uuid-2c990270-d140-4d92-8bca-629e21926535"/>
      </div>
      <div LABEL="schema files"/>
      <div LABEL="content files">
        <fptr FILEID="uuid-cf9818bb-567b-44ee-88d8-60a1420feae3"/>
        <fptr FILEID="uuid-3b0e4dcb-727a-44d1-af24-d35676b02bed"/>
      </div>
    </div>
  </structMap>
</mets>
```

6.3 Appendix C - PREMIS.xml describing events on package level

```
<premis xmlns="info:lc/xmlns/premis-v2" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" version="2.0" xsi:schemaLocation="info:lc/xmlns/premis-v2
```

```
     ../../schemas/premis-v2-2.xsd">
   <object xmlID="uuid-187f239d-c080-4a7f-936d-b35cec4e8ef7" xsi:type="
      representation">
     <objectIdentifier>
       <objectIdentifierType>repository</objectIdentifierType>
       <objectIdentifierValue>urn:uuid:7ff70669-73a0-4551-ad5b-12ed9b229e38</
    objectIdentifierValue>
     </objectIdentifier>
   </object>
   <event>
     <eventIdentifier>
       <eventIdentifierType>local</eventIdentifierType>
       <eventIdentifierValue>IDc5d159d7-2df0-4efe-b07b-559fac4bdc27</
    eventIdentifierValue>
     </eventIdentifier>
     <eventType>SIP Delivery Validation</eventType>
     <eventDateTime>2016-12-14T09:14:04</eventDateTime>
     <eventOutcomeInformation>
       <eventOutcome>success</eventOutcome>
     </eventOutcomeInformation>
     <linkingAgentIdentifier>
       <linkingAgentIdentifierType>software</linkingAgentIdentifierType>
       <linkingAgentIdentifierValue>E-ARK Web 0.9.4 (task: SIPDeliveryValidation)
    </linkingAgentIdentifierValue>
     </linkingAgentIdentifier>
     <linkingObjectIdentifier>
       <linkingObjectIdentifierType>repository</linkingObjectIdentifierType>
       <linkingObjectIdentifierValue>urn:uuid:7ff70669-73a0-4551-ad5b-12
    ed9b229e38</linkingObjectIdentifierValue>
     </linkingObjectIdentifier>
   </event>
   <agent>
     <agentIdentifier>
       <agentIdentifierType>LOCAL</agentIdentifierType>
       <agentIdentifierValue>E-ARK Web 0.9.4</agentIdentifierValue>
     </agentIdentifier>
     <agentName>E-ARK Web</agentName>
     <agentType>Software</agentType>
   </agent>
 </premis>
```

6.4 Appendix D - PREMIS.xml describing migration events (representation level)

```
<?xml version='1.0' encoding='UTF-8'?>
<premis xmlns="info:lc/xmlns/premis-v2" xmlns:xsi="http://www.w3.org/2001/
    XMLSchema-instance" version="2.0" xsi:schemaLocation="info:lc/xmlns/premis-v2
     ../../schemas/premis-v2-2.xsd">
  <object xsi:type="representation" xmlID="IDd61654d8-44dd-4dee-872c-89
      cf1ad240bf">
    <objectIdentifier>
      <objectIdentifierType>repository</objectIdentifierType>
      <objectIdentifierValue>IDd61654d8-44dd-4dee-872c-89cf1ad240bf</
    objectIdentifierValue>
    </objectIdentifier>
  </object>
  <object xsi:type="file" xmlID="ID9f65a8bd-5128-4830-a28b-0acb4455e128">
    <objectIdentifier>
      <objectIdentifierType>filepath</objectIdentifierType>
      <objectIdentifierValue>./data/OAIS_Wikipedia_Article.pdf</
    objectIdentifierValue>
    </objectIdentifier>
    <objectCharacteristics>
      <compositionLevel>0</compositionLevel>
      <fixity>
        <messageDigestAlgorithm>SHA-256</messageDigestAlgorithm>
        <messageDigest>
    a0d26c309030408b2a6618805c3747d9d04599f70051be9630d948cecaed3a0e</
    messageDigest>
        <messageDigestOriginator>hashlib</messageDigestOriginator>
      </fixity>
      <size>4667731</size>
      <format>
        <formatRegistry>
          <formatRegistryName>PRONOM</formatRegistryName>
          <formatRegistryKey>fmt/276</formatRegistryKey>
          <formatRegistryRole>identification</formatRegistryRole>
        </formatRegistry>
      </format>
    </objectCharacteristics>
    <relationship>
      <relationshipType>derivation</relationshipType>
      <relationshipSubType>has source</relationshipSubType>
      <relatedObjectIdentification>
        <relatedObjectIdentifierType>filepath</relatedObjectIdentifierType>
        <relatedObjectIdentifierValue>./../0910ba24-f328-4083-a05f-cce0cb3eb49f/
    data/OAIS_Wikipedia_Article.pdf</relatedObjectIdentifierValue>
        <relatedObjectSequence>0</relatedObjectSequence>
      </relatedObjectIdentification>
      <relatedEventIdentification>
        <relatedEventIdentifierType>local</relatedEventIdentifierType>
        <relatedEventIdentifierValue>ID1656ae4a-9f1b-43a5-9e56-ddaef284ec71</
    relatedEventIdentifierValue>
        <relatedEventSequence>1</relatedEventSequence>
      </relatedEventIdentification>
    </relationship>
  </object>
  <event>
    <eventIdentifier>
      <eventIdentifierType>local</eventIdentifierType>
      <eventIdentifierValue>ID1656ae4a-9f1b-43a5-9e56-ddaef284ec71</
    eventIdentifierValue>
    </eventIdentifier>
    <eventType>migration</eventType>
    <eventDateTime>2021-03-15T16:33:17</eventDateTime>
    <eventOutcomeInformation>
      <eventOutcome>success</eventOutcome>
    </eventOutcomeInformation>
    <linkingAgentIdentifier>
      <linkingAgentIdentifierType>software</linkingAgentIdentifierType>
      <linkingAgentIdentifierValue>GPL Ghostscript 9.26 (2018-11-20)</
    linkingAgentIdentifierValue>
    </linkingAgentIdentifier>
    <linkingObjectIdentifier>
      <linkingObjectIdentifierType>filepath</linkingObjectIdentifierType>
      <linkingObjectIdentifierValue>./data/OAIS_Wikipedia_Article.pdf</
    linkingObjectIdentifierValue>
    </linkingObjectIdentifier>
  </event>
  <agent>
    <agentIdentifier>
      <agentIdentifierType>LOCAL</agentIdentifierType>
      <agentIdentifierValue>Premis Generator</agentIdentifierValue>
    </agentIdentifier>
    <agentName>Premis Generator</agentName>
    <agentType>Software</agentType>
  </agent>
  <agent>
```

```
    <agentIdentifier>
      <agentIdentifierType>LOCAL</agentIdentifierType>
      <agentIdentifierValue>GPL Ghostscript 9.26 (2018-11-20)
</agentIdentifierValue>
    </agentIdentifier>
    <agentName>GPL Ghostscript 9.26 (2018-11-20)
</agentName>
    <agentType>Software</agentType>
  </agent>
</premis>
```

6.5 Appendix E - Naming scheme examples

6.5.1 Migrating a representation to a new version

In the example shown in Figure 8, a single physical container file includes metadata and two representations, namely representation R1 (JPEG2000) and representation R2 (PNG). Representation R1 is to be migrated to R1.1 (TIFF). Representation R1 (JPEG2000) is migrated to a new representation R1.1 (TIFF). The "version" suffix of the physical container file name is incremented. The structural information (`STRUCTURE`) in the new version of the physical container file is updated so that it references the new version R1.1. Note that a copy of representation R2 (PNG) is created so that this representation is stored redundantly.

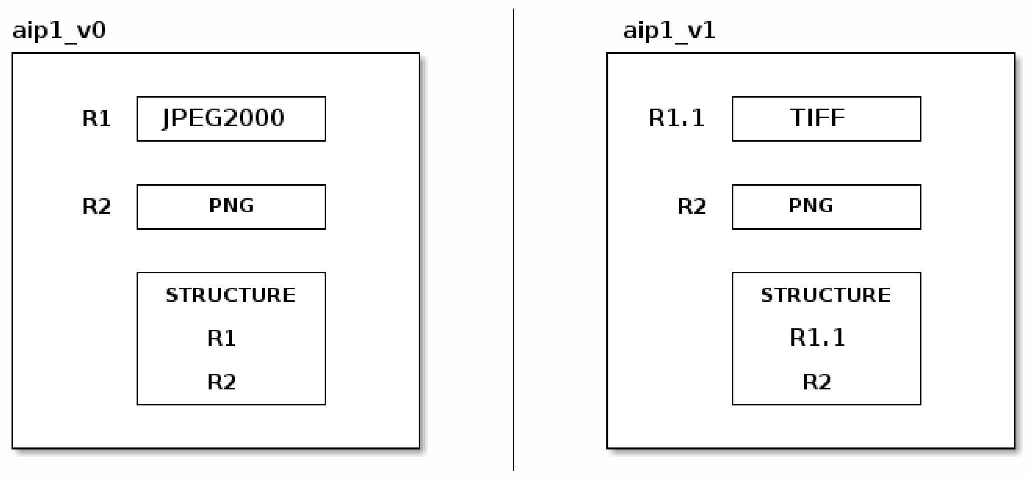

Figure 8: Migrating a representation to a new version.

6.5.2 Migrating a representation to a new version with segmented packages

In the example shown in Figure 9, an AIP is segmented. There are two physical container with the representations as child packages and one physical container file as the parent package which holds the root METS file. This means that representation R1 (JPEG2000) and representation R2 (PNG) are stored in two separate container files.

Representation R1 is migrated to R1.1 (TIFF). A new version named `aip1_v1_b1` of the `aip1_v0_b1` container file is created. The physical container file `aip1_v0_b2` remains unchanged.

The parent physical container file `aip1_v0` which holds the references to child packages is also updated to the new version `aip1_v1`.

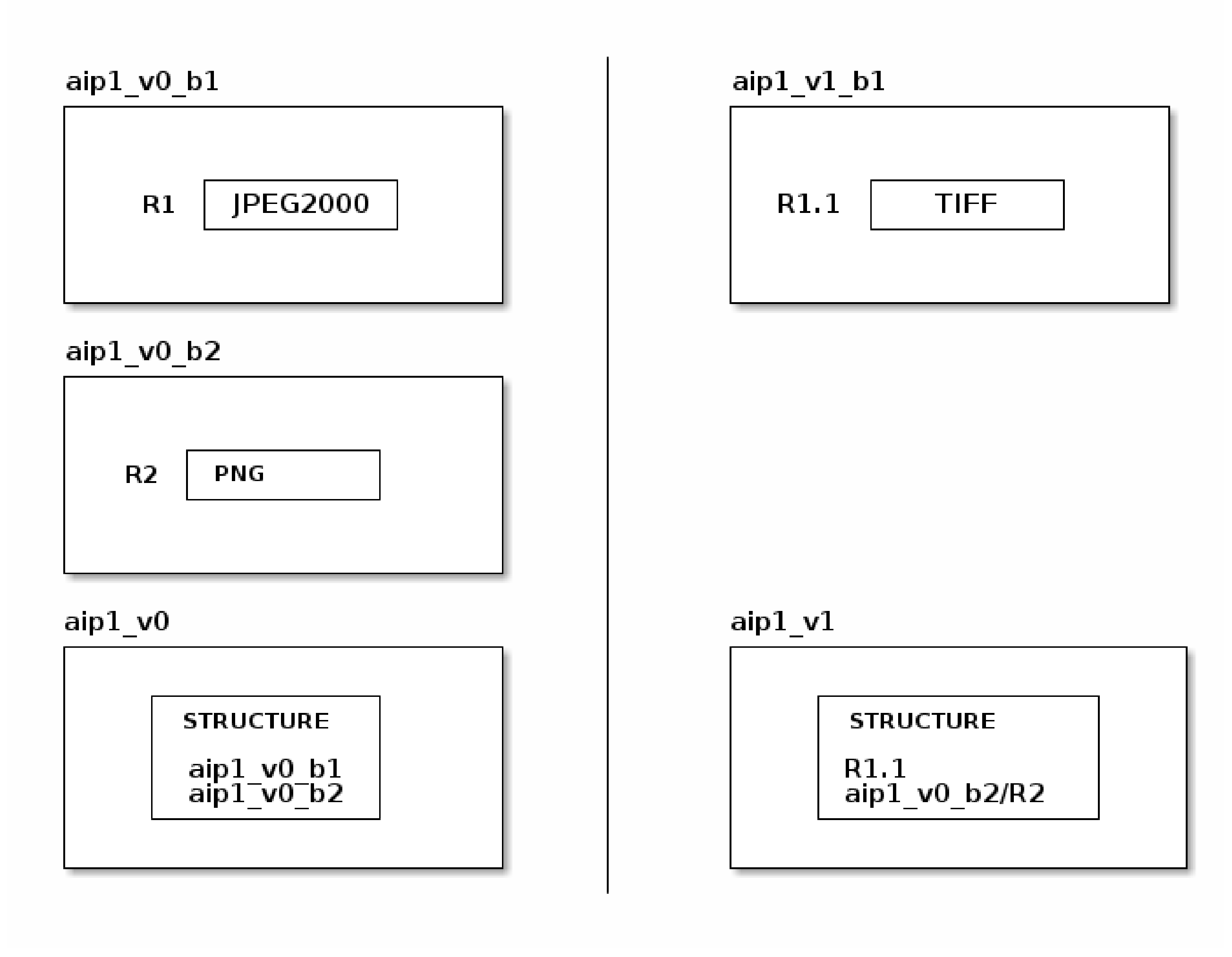

Figure 9: Migrating a representation to a new version with segmented packages.

6.5.3 Migrating a representation using a differential package

In the example shown in Figure 10, a single physical container file includes metadata and two representations, namely representation R1 (JPEG2000) and representation R2 (PNG). Representation R1 is to be migrated to R1.1 (TIFF).

Representation R1 is migrated to a differential package which only stores the representation and structural information which was modified.

Note that the version number is not incremented for the differential package. The suffix `d1` indicates that the physical container file is a differential physical container file which relates to the previous complete state which is stored in the physical container file `aip1_v0`. The differential physical container file is incomplete and needs to be consolidated into a new consolidated version `aip1_v1` of the physical container file which is complete.

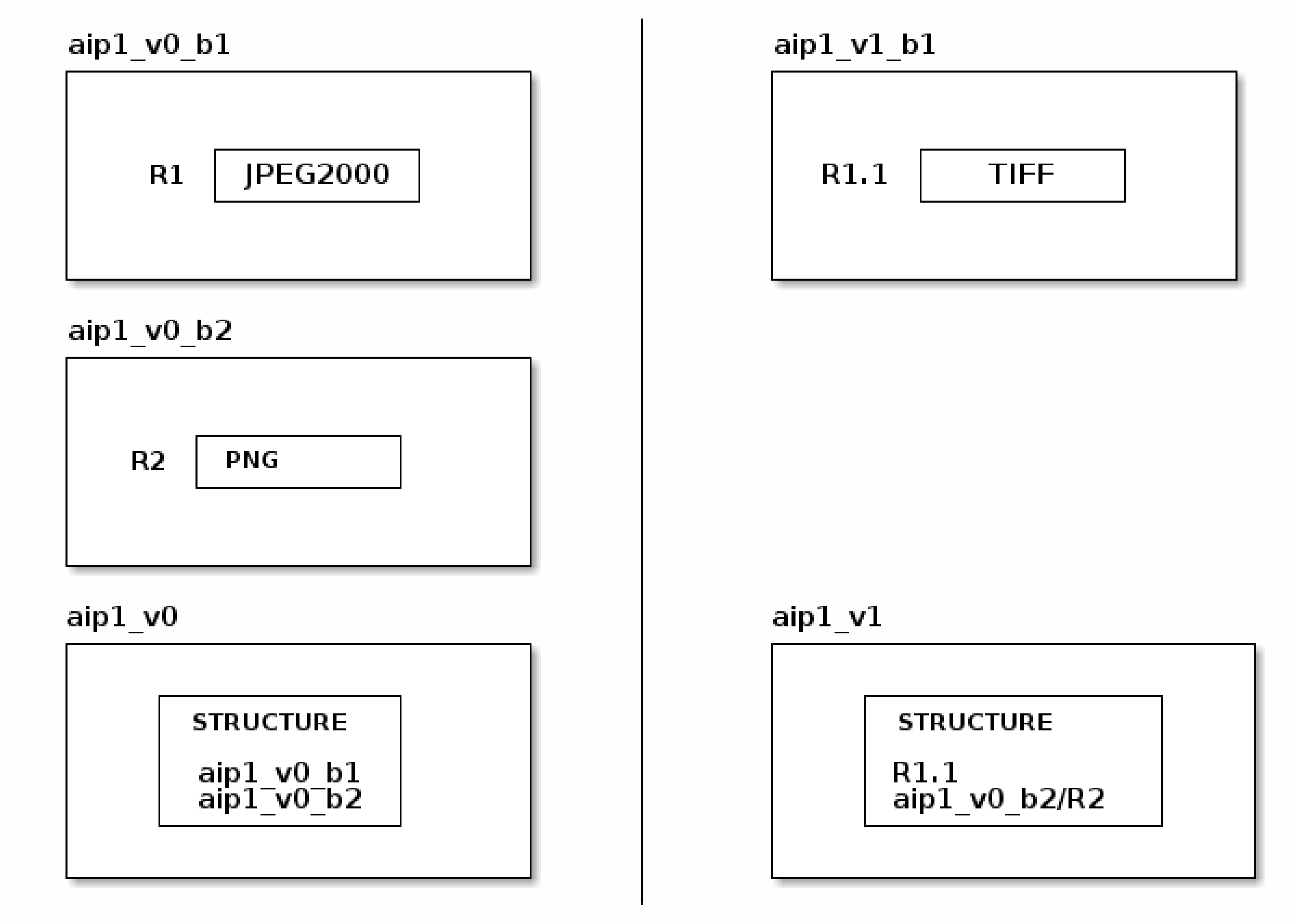

Figure 10: Migrating a representation using a differential package.

References

Bredenberg, Karin, Bjorn Skog, Anders Bo Nielsen, Kathrine Hougaard Edsen Johansen, Alex Thirifays, Sven Schlarb, Andrew Wilson, et al. 2018. *Common Specification for Information Packages (Csip)*. *ERCIM News*. 2.0.0-DRAFT ed. Digital Information LifeCycle Interoperability Standard Board (DILCIS Board). http://earkcsip.dilcis.eu.

Faria, Luis, Miguel Ferreira, Jan Rörden, and Sven Schlarb. 2017. *D4.4 Sip-Aip Conversion Component*. http://www.eark-project.com/resources/project-deliverables/89-d44.

OAIS. 2012. *Reference Model for an Open Archival Information System*. CCSDS 650.0-M-2 (Magenta Book). CCSDS - Consultative Committee for Space Data Systems. http://public.ccsds.org/publications/archive/650x0b1.pdf.

PREMIS. 2017. *PREMIS Data Dictionary for Preservation Metadata, Version 3.0*. The Library of Congress. https://www.loc.gov/standards/premis/v3/index.html.

Postface

I. Authors

Name	Organisation
Karin Bredenberg	National Archives of Sweden
Luis Faria	Keep Solutions
Miguel Ferreira	Keep Solutions
Anders Bo Nielsen	Danish National Archives
Jan Rörden	Austrian Institute of Technology
Sven Schlarb	Austrian Institute of Technology
Carl Wilson	Open Preservation Foundation

II. Revision History

Revision No.	Date	Authors(s)	Description
0.1	20.09.2016	Sven Schlarb Jan Rörden	First draft based on E-ARK deliverable D4.3.
0.2	15.10.2016	Miguel Ferreira Luis Faria	Comments, Contribution
0.9	20.12.2016	Sven Schlarb	Provided for internal review (E-ARK deliverable D4.4)
0.9.1	06.01.2017	Andrew Wilson	Comments and language review

Revision No.	Date	Authors(s)	Description
0.9.2	13.01.2017	Kuldar Aas	Comments
1.0	27.01.2017	Sven Schlarb Jan Rörden	Address review comments and language; final changes.
2.0-DRAFT	12.12.2018	Sven Schlarb Carl Wilson	Migration to markdown, review
2.0.0	15.05.2019	Carl Wilson Sven Schlarb	Version 2.0.0
2.0.1	09.09.2019	Carl Wilson	Site structure and PDF layout
2.0.4	12.06.2020	K. Bredenberg C.Wilson J. Kaminski	Preface text and output display update
2.1.0	15.10.2021	Sven Schlarb	<Summary of 2.1.0 HERE>

III Acknowledgements

The E-ARK Archival Information Package (DIP) Specification was first developed within the E-ARK project in 2014 – 2017. E-ARK was an EC-funded pilot action project in the Competitiveness and Innovation Programme 2007- 2013, Grant Agreement no. 620998 under the Policy Support Programme.

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

IV Contact & Feedback

The E-ARK AIP specification is maintained by the Digital Information LifeCycle Interoperability Standard Board (DILCIS Board). For further information about the DILCIS Board or feedback on the current document please consult the website http://www.dilcis.eu/ or https://github.com/dilcisboard or contact us at info@dilcis.eu.