SMPTE ST 429-9:2014
Revision of
SMPTE 429-9-2007

SMPTE STANDARD

D-Cinema Packaging — Asset Mapping and File Segmentation

Table of Contents

Page

Foreword 2
Intellectual Property 2
1 Scope 3
2 Conformance Notation 3
3 Normative References 3
4 Overview 4
4.1 Ingesting a Mapped File Set (Informative) 5
4.2 XML File Structure 5
5 AssetMap Structure 6
5.1 Id 6
5.2 AnnotationText [optional] 7
5.3 Creator 7
5.4 VolumeCount 7
5.5 IssueDate 7
5.6 Issuer 7
5.7 AssetList 7
6 Asset Structure 7
6.1 Id 8
6.2 AnnotationText [optional] 8
6.3 PackingList [optional] 8
6.4 ChunkList 8
7 Chunk Structure 8
7.1 Path 9
7.2 VolumeIndex [optional] 9
7.3 Offset [optional] 9
7.4 Length [optional] 9
8 VolumeIndex Structure 9

Approved
November 24, 2014

9 Media-Specific Constraints 10
9.1 Mapped File Set Definition 10
9.2 Path URI 10
9.3 XML Encoding 10
9.4 Asset Map Location 10
10 Asset Map Sample (Informative) 11
11 XML Schema 12
Annex A Basic Map Profile v2 (Normative) 14
A.1 Mapped File Set Definition 14
A.2 Path 14
A.3 XML Encoding 14
A.4 Asset Map Location 14
A.5 Example Ingest Algorithm (Informative) 15
Annex B XML Diagram Legend (Informative) 16
B.1 Element Symbols 16
B.2 Model Symbols ("Compositors") 17
B.3 Types 17
B.4 Model Groups and References 18

Foreword

SMPTE (the Society of Motion Picture and Television Engineers) is an internationally-recognized standards developing organization. Headquartered and incorporated in the United States of America, SMPTE has members in over 80 countries on six continents. SMPTE's Engineering Documents, including Standards, Recommended Practices and Engineering Guidelines, are prepared by SMPTE's Technology Committees. Participation in these Committees is open to all with a bona fide interest in their work. SMPTE cooperates closely with other standards-developing organizations, including ISO, IEC and ITU.

SMPTE Engineering Documents are drafted in accordance with the rules given in its Standards Operations Manual.

SMPTE ST 429-9 was prepared by Technology Committee 21DC.

Intellectual Property

At the time of publication no notice had been received by SMPTE claiming patent rights essential to the implementation of this Standard. However, attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. SMPTE shall not be held responsible for identifying any or all such patent rights.

1 Scope

This document specifies a generic method of mapping asset identifiers (UUID) to locations (URI). An application of the method to common hierarchical filesystems, the Basic Map Profile v2, is specified.

Note: Earlier versions of this specification allowed assets to be split among multiple volumes. This functionality was not found to be useful in practice and is removed from this version. The schema remains however unchanged to minimize impact on implementations.

2 Conformance Notation

Normative text is text that describes elements of the design that are indispensable or contains the conformance language keywords: "shall", "should", or "may". Informative text is text that is potentially helpful to the user, but not indispensable, and can be removed, changed, or added editorially without affecting interoperability. Informative text does not contain any conformance keywords.

All text in this document is, by default, normative, except: the Introduction, any section explicitly labeled as "Informative" or individual paragraphs that start with "Note:"

The keywords "shall" and "shall not" indicate requirements strictly to be followed in order to conform to the document and from which no deviation is permitted.

The keywords, "should" and "should not" indicate that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain possibility or course of action is deprecated but not prohibited.

The keywords "may" and "need not" indicate courses of action permissible within the limits of the document. The keyword “reserved” indicates a provision that is not defined at this time, shall not be used, and may be defined in the future. The keyword “forbidden” indicates “reserved” and in addition indicates that the provision will never be defined in the future.

3 Normative References

The following standards contain provisions which, through reference in this text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent edition of the standards indicated below.

SMPTE ST 429-8:2007, D-Cinema Packaging — Packing List

Internet Engineering Task Force (IETF) (2005, January). RFC 3986 — Uniform Resource Identifiers (URI): Generic Syntax

Internet Engineering Task Force (IETF) (2005, July). RFC 4122 — A Universally Unique Identifier (UUID) URN Namespace.

World Wide Web Consortium (W3C) (2004, February 4). Extensible Markup Language (XML) 1.0 (Third Edition).

World Wide Web Consortium (W3C) (2004, October 28). XML Schema Part 1: Structures (Second Edition).

World Wide Web Consortium (W3C) (2004, October 28). XML Schema Part 2: Datatypes (Second Edition).

World Wide Web Consortium (W3C) (2009, December 8). Namespaces in XML,

4 Overview

The packing list [SMPTE ST 429-8] specifies the contents of a distribution package.

The package and its Packing List are specified independently of the underlying media, e.g. hard disk partition. The Packing List contains the UUID identifiers of the assets in the Package, not the location of those assets on the underlying media. The Asset Map provides the link between the asset identifier UUIDs and the files containing the assets. This layered approach has two distinct advantages. First, it simplifies specification of the Packing List by removing dependence on past, current or future media formats and constraints. Second, it allows the package and its underlying assets to be created once and mapped onto various target media as circumstances require.

The Asset Map structure, depicted in Figure 1, provides a mapping from the Packing List Id and asset Ids to actual locations within a Mapped File Set, which consists of the Asset Map and all referenced assets. An Asset Map can contain mappings for more than one package, but does not contain a partial mapping for a particular package. The Asset Map provides special identification for Packing List assets so that they can be located without scanning all assets within the package.

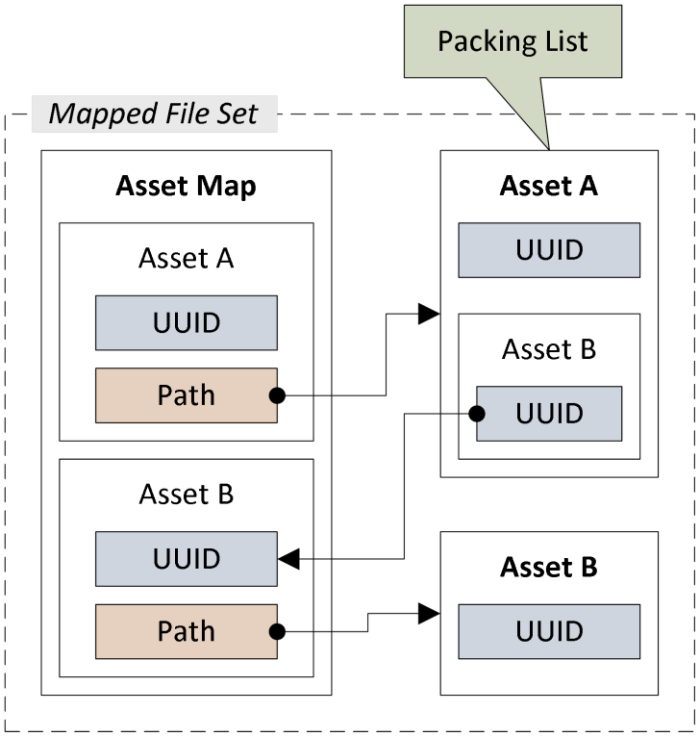


Figure 1 – Relationship between the Packing List and Asset Map structures

The definition of Mapped File Set and the means of locating the Asset Map are defined by a normative Map Profile (see Section 9) for that media type.

Annex A specifies a Map Profile for use with common hierarchical filesystems.

4.1 Ingesting a Mapped File Set (Informative)

The process of reading a package by an implementation is termed ingest. A system can provide a simple or elaborate ingest control interface — allowing coarse or fine-grained selection of items to be ingested — but the basic process will typically be the same.

As illustrated using pseudo-code in Figure 2, a Mapped File Set is ingested by first opening its Asset Map document. The means of identifying the Asset Map on the Mapped File Set is defined by the normative Map Profile (see Section 9) for that media type. The Asset Map is used to locate the Packing List(s) which detail the contents of the available package(s). Assets are chosen from the Packing List(s), and the selected assets are ingested.

```
Ingest Mapped File Set:
  open the Asset Map

  for each Packing List in the Asset Map
    open the Packing List

    for each Asset in the Packing List
      if this Asset is wanted
        locate the corresponding Asset structure in the Asset Map

        read the Asset and test the message digest
```

Figure 2 – Example ingest process

4.2 XML File Structure

The structures defined in this document are represented using the Extensible Markup Language (XML) [XML 1.0], and specified using XML Schema [XML Schema Part 1: Structures] and [XML Schema Part 2: Datatypes]. This specification shall be associated with a unique XML namespace name [Namespaces in XML]. The namespace name shall be the string value "`http://www.smpte-ra.org/schemas/429-9/2007/AM`". This namespace name conveys both structural and semantic version information, but does not serve the purpose of a traditional version number field.

Table 1 lists the XML namespace names used in this specification. Namespace names are represented as Uniform Resource Identifier (URI) values [RFC 3986][1].

[1] Readers unfamiliar with URI values as XML namespace names should be aware that although a URI value begins with a “method” element (“http” in this case), the value is designed primarily to be a unique string and does not necessarily correspond to an actual on-line resource. Applications implementing this standard should not attempt to resolve URI values on-line.

Table 1 – XML Namespaces

Qualifier	*URI*
am	`http://www.smpte-ra.org/schemas/429-9/2007/AM`
xs	`http://www.w3.org/2001/XMLSchema`

The URIs found in Table 1 are normative. The namespace qualifier values (also called namespace prefixes in XML jargon) used in Table 1 and elsewhere in this document, namely "am" and "xs", are not normative. Specifically, they may be replaced in instance documents by any XML compliant namespace prefix. In other words, implementations shall expect any arbitrary XML compliant namespace prefix value that is associated with a URI from Table 1.

5 AssetMap Structure

An Asset Map document is an XML document with an `AssetMap` element as its root element. The elements and structure of an AssetMap element are illustrated in Figure 3 and defined in the subsections that follow.

An Asset Map may contain mappings for more than one package, but shall not contain a partial mapping for a particular package, i.e. a package shall not span multiple distinct Asset Maps.

A Mapped File Set shall contain exactly one Asset Map.

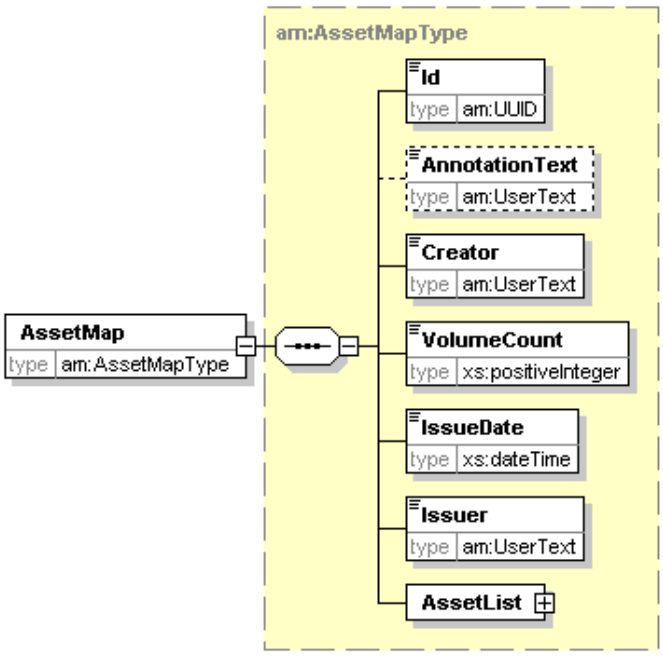


Figure 3 – AssetMap structure

5.1 Id

The `Id` element uniquely identifies the Asset Map structure. It is encoded as a `urn:UUID` [RFC 4122].

5.2 AnnotationText [optional]

The `AnnotationText` element is a free-form, human-readable annotation describing the Asset Map. It is meant strictly as a display hint to the user. The optional `language` attribute is an `xs:language` language code and indicates the language used. If the `language` attribute is not present, the default value `en` shall be used.

5.3 Creator

The `Creator` element is a free-form, human-readable annotation describing the system (hardware/software) that was used to create the Asset Map for distribution. It is meant strictly as a display hint to the user. The optional `language` attribute is an `xs:language` language code and indicates the language used. If the `language` attribute is not present, the default value `en` shall be used.

5.4 VolumeCount

The `VolumeCount` element shall be 1 (one).

Note: Earlier versions of this specification included support for distributing Asset Maps across multiple storage volumes.

5.5 IssueDate

The `IssueDate` element indicates the time and date at which the Asset Map was issued. It may be displayed to the user.

5.6 Issuer

The `Issuer` element is a free-form, human-readable annotation describing the person or company who has created the Asset Map for distribution. It is meant strictly as a display hint to the user. The optional `language` attribute is an `xs:language` language code and indicates the language used. If the `language` attribute is not present, the default value `en` shall be used.

5.7 AssetList

The `AssetList` element contains a list of `Asset` elements. The structure of the `Asset` element is described in the following section.

6 Asset Structure

The `Asset` element is used to represent assets within the `AssetList` element of an `AssetMap`. The `Asset` element is of the type `AssetType`, illustrated in Figure 4 and defined in the subsections that follow.

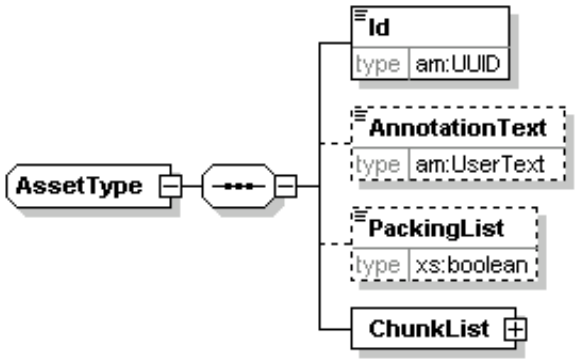


Figure 4 – Asset structure

6.1 Id

The `Id` element uniquely identifies the asset for management purposes. It is encoded as a `urn:UUID` [RFC 4122].

6.2 AnnotationText [optional]

The `AnnotationText` element is a free-form, human-readable annotation describing the asset. It is meant strictly as a display hint to the user. The optional `language` attribute is an `xs:language` language code and indicates the language used. If the `language` attribute is not present, the default value `en` shall be used.

6.3 PackingList [optional]

The `PackingList` element indicates whether the asset is a Packing List [SMPTE ST 429-8]. It is encoded as an `xs:boolean`. If the asset is a Packing List, the element shall be present and the value shall be `true`, otherwise the element shall be absent or its value shall be `false`. The default value is `false`.

6.4 ChunkList

The `ChunkList` element shall contain one `Chunk` element, which provides information on the location of the asset.

The structure of the `Chunk` element is described in the following section.

Note: Earlier versions of this specification included support for segmenting assets across multiple chunks. This version requires that the asset is contained in a single chunk.

7 Chunk Structure

The `Chunk` element is used to represent chunks within the `ChunkList` element of an `Asset`. The `Chunk` element is of the type `ChunkType`, illustrated in Figure 5 and defined in the subsections that follow.

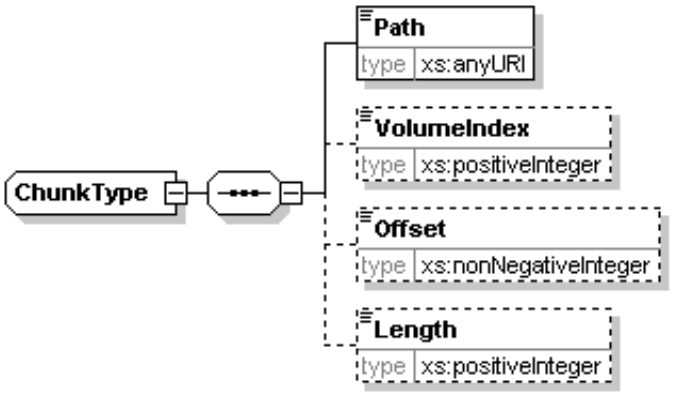


Figure 5 – Chunk structure

7.1 Path

The `Path` element indicates the complete path for the Chunk, represented as a URI per [RFC 3986]. Its semantics and format are delivery-medium dependent, and constrained by each Map Profile (see Section 9). The value is encoded as an `xs:anyURI`.

Note: Annex A presents a basic Map Profile.

7.2 VolumeIndex [optional]

The `VolumeIndex` element shall be equal to 1 or absent.

Note: Earlier versions of this specification included support for distributing Asset Maps across multiple storage volumes

7.3 Offset [optional]

The `Offset` element shall be equal to 0 or absent.

Note: Earlier versions of this specification supported segmentation of assets in multiple chunks to address filesystems with file size limits.

7.4 Length [optional]

The `Length` element identifies the length, in bytes, of the chunk. It shall be absent, or equal to the length in bytes of the asset.

Note: Earlier versions of this specification supported segmentation of assets in multiple chunks to address filesystems with file size limits.

8 VolumeIndex Structure

The VolumeIndex structure is not used. Both the `VolumeIndex` global element (but not the `VolumeIndex` element of the `ChuckType` type) and the `VolumeIndexType` global type are deprecated accordingly.

Note: Earlier versions of this specification included support for distributing Asset Maps across multiple storage volumes.

9 Media-Specific Constraints

The structures defined above do not alone provide sufficient specification for mapping a package onto a particular media. Many details must be defined to form a complete, standardized mapping, including the format and access method of the media and the paths of the Asset Map document on an instance of that media.

A Map Profile document specifies, or references specifications of, a file storage system and the electronic and/or logical means of retrieving files from it. A Map Profile specifies the manner by which a Mapped File Set shall be retrieved from the specified media, including any necessary constraints on the use of the media and the structures in this document.

A Map Profile document shall normatively reference this document and shall follow the provisions of this document except where explicitly noted.

A basic, normative Map Profile for random-access filesystems is given in Annex A.

9.1 Mapped File Set Definition

The Map Profile shall explicitly define the term Mapped File Set in the context of the particular media being addressed. For example, a disk-based mapping may define a Mapped File Set as a set of constraints on an existing filesystem and disk interface, while a network mapping might specify a host addressing method and file retrieval protocol.

9.2 Path URI

The Map Profile may specify constraints on the value of the Path element.

For instance, the Map Profile can constrain the URI scheme, path length, directory depth and directory separator, character set, etc.

9.3 XML Encoding

If applicable, the Map Profile should specify an XML encoding method (e.g. UTF-8) for Asset Map documents.

9.4 Asset Map Location

The Map Profile shall specify means of locating Asset Map structures.

10 Asset Map Sample (Informative)

```
<?xml version="1.0" encoding="UTF-8"?>
<am:AssetMap xmlns:am="http://www.smpte-ra.org/schemas/429-9/2007/AM">
   <am:Id>urn:uuid:3a64e8ac-80ed-4d70-b348-ffc2d241d76e</am:Id>
   <am:AnnotationText>Magic Packager 3.3</am:AnnotationText>
   <am:Creator>Asset Mapper 2.0</am:Creator>
   <am:VolumeCount>1</am:VolumeCount>
   <am:IssueDate>2014-02-17T07:30:46-00:00</am:IssueDate>
   <am:Issuer>Analog Labs</am:Issuer>
   <am:AssetList>
      <am:Asset>
         <am:Id>urn:uuid:b4ab6bd6-a8ba-4a9a-9d0a-00668792ae20</am:Id>
         <am:AnnotationText>Packing List for The Jazz Singer</am:AnnotationText>
         <am:PackingList>true</am:PackingList>
         <am:ChunkList>
            <am:Chunk>
               <am:Path>JazzSingerEN-0003278645.pkl.xml</am:Path>
            </am:Chunk>
         </am:ChunkList>
      </am:Asset>
      <am:Asset>
         <am:Id>urn:uuid:ff3209c7-f27e-444f-9dce-330397a5dd42</am:Id>
         <am:AnnotationText>Picture Track File for Reel #1 for The Jazz
                           Singer</am:AnnotationText>
         <am:ChunkList>
            <am:Chunk>
               <am:Path>JS-235-2K-R1.ptrk.mxf.aa</am:Path>
            </am:Chunk>
         </am:ChunkList>
      </am:Asset>
   </am:AssetList>
</am:AssetMap>
```

11 XML Schema

The XML Schema document presented in this section normatively defines the structure of the Asset Map and Volume Index using XSDL. While this schema is intended to faithfully represent the structure presented in the normative prose portions (Sections 4 through 8) of this document, conflicts in definition may occur. In the event of such a conflict, the normative prose shall be the authoritative expression of the standard.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.smpte-ra.org/schemas/429-9/2007/AM"
   xmlns:am="http://www.smpte-ra.org/schemas/429-9/2007/AM"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xs:simpleType name="UUID">
      <xs:restriction base="xs:anyURI">
         <xs:pattern value="urn:uuid:[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-
                            [0-9a-fA-F]{12}"/>
      </xs:restriction>
   </xs:simpleType>

   <xs:complexType name="UserText">
      <xs:simpleContent>
         <xs:extension base="xs:string">
            <xs:attribute name="language" type="xs:language" use="optional" default="en"/>
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>

   <xs:complexType name="ChunkType">
      <xs:sequence>
         <xs:element name="Path" type="xs:anyURI"/>
         <xs:element name="VolumeIndex" type="xs:positiveInteger" minOccurs="0"/>
         <xs:element name="Offset" type="xs:nonNegativeInteger" minOccurs="0"/>
         <xs:element name="Length" type="xs:positiveInteger" minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>

   <xs:complexType name="AssetType">
      <xs:sequence>
         <xs:element name="Id" type="am:UUID"/>
         <xs:element name="AnnotationText" type="am:UserText" minOccurs="0"/>
         <xs:element name="PackingList" type="xs:boolean" minOccurs="0"/>
         <xs:element name="ChunkList">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="Chunk" type="am:ChunkType" maxOccurs="unbounded"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
 <!-- continued next page... -->
```

```
<!-- Asset Map schema continued -->
   <xs:complexType name="AssetMapType">
      <xs:sequence>
         <xs:element name="Id" type="am:UUID"/>
         <xs:element name="AnnotationText" type="am:UserText" minOccurs="0"/>
         <xs:element name="Creator" type="am:UserText"/>
         <xs:element name="VolumeCount" type="xs:positiveInteger"/>
         <xs:element name="IssueDate" type="xs:dateTime"/>
         <xs:element name="Issuer" type="am:UserText"/>
         <xs:element name="AssetList">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="Asset" type="am:AssetType" maxOccurs="unbounded"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>

   <xs:element name="AssetMap" type="am:AssetMapType"/>

   <xs:complexType name="VolumeIndexType">
      <xs:sequence>
         <xs:element name="Index" type="xs:positiveInteger"/>
      </xs:sequence>
   </xs:complexType>

   <xs:element name="VolumeIndex" type="am:VolumeIndexType"/>
</xs:schema>
```

Annex A Basic Map Profile v2 (Normative)

This annex defines a Map Profile for mapping one or more Mapped File Sets onto a hierarchical, random-access filesystem which supports long path names and files larger than 4 GB. This Map Profile is intended to be used with direct attached storage. Support for other access methods, e.g. FTP, is not specified here.

The Basic Map Profile defined in earlier versions of this specification is deprecated.

A.1 Mapped File Set Definition

A Mapped File Set is the tree rooted at the directory containing an Asset Map file.

A.2 Path

Each Path element value shall be a relative-path reference as specified in RFC 3986. No query or fragment component shall be present.

Given a Path element in an Asset Map, the relative-path reference shall be resolved, as specified in RFC 3986, relative to a Base URI consisting of the location of the Asset Map.

EXAMPLE 1: Given an Asset Map URI equal to `/mnt/disk1/package_1/ASSETMAP.xml`, the URI of an Asset with a Path element equal to `composition_1/2423.mxf` is `/mnt/disk1/package_1/composition_1/2423.mxf`.

Every location traversed by this resolved relative-path reference shall be within the tree rooted at the directory containing the Asset Map, i.e. within the corresponding Mapped File Set.

EXAMPLE 2: For an Asset Map located at `/foo/ASSETMAP.xml`, the Path element value `bar/../../foo/bar/Image.mxf` is forbidden since the location `bar/../..` is outside the tree rooted at `/foo`.

If offered by the filesystem, hard links may be used to avoid duplication of information if the same underlying file is referenced by multiple Asset Maps on the same filesystem. Even if offered by the filesystem, symbolic links should not be used however as copy or presence of the referenced file is not always guaranteed.

Note 1: A symbolic link typically refers to a filesystem node that references another node, which may not exist at all or exist on a different medium, in the form of a path. In contrast, a hard link refers to a filesystem node that references the same exact contents as one or more other nodes.

Each path segment, as specified in IETF RFC 3986, shall consist of characters from the set a-z, A-Z, 0-9, “-“ (dash), “_” (underscore) and “.” (period). No segment shall have more than 100 characters, and the value of the Path element shall not exceed 100 characters in length. A Path element value shall have no more than 10 segments. The Path element value shall preserve case (the path and the filename on the filesystem shall have identical case). No two paths in an Asset Map shall have identical value, regardless of case.

Note 2: A relative-path reference cannot start with a "/" character.

Note 3: "." and ".." path segments are allowed.

A.3 XML Encoding

The Asset Map document shall be encoded using the UTF-8 encoding [XML 1.0].

A.4 Asset Map Location

The filesystem shall contain either:

- a single Asset Map document in root directory of the filesystem; or
- one or more Asset Map documents, each within a directory whose immediate parent is the root directory, i.e. a top-level directory. The length of the name of the directory containing the Asset Map document shall be no more than 100 characters.

For example, Asset Map documents will not be present both (i) in the root directory of the filesystem and (ii) within directories whose immediate parent directory is the root directory.

Each Asset Map document shall be a file named "ASSETMAP.xml".

Note: Similarly named files, e.g. "ASSETMAP" can be present, but are not considered in the context of this specification, i.e. they are ignored.

A.5 Example Ingest Algorithm (Informative)

The following is an example ingest process for a filesystem organized according to the Basic Map Profile v2. The Ingest Mapped File Set algorithm is defined in Section 4.1.

```
Ingest Basic Map Profile V2 Filesystem:
  if root directory contains a file named "ASSETMAP.xml"
    Ingest Mapped File Set in root directory
  else
    for each top level directory of root directory:
      if top level directory contains a file named "ASSETMAP.xml"
        Ingest Mapped File Set in top level directory
```

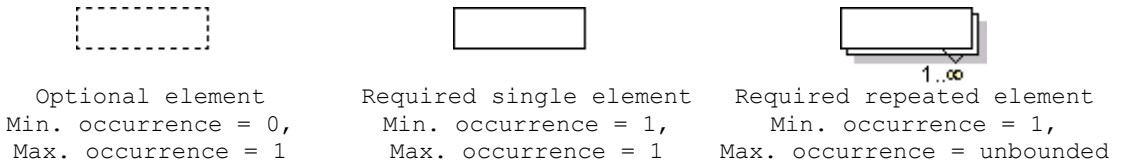
Figure A.1 – Example ingest process for a filesystem organized according to the Basic Map Profile V2

Annex B XML Diagram Legend (Informative)

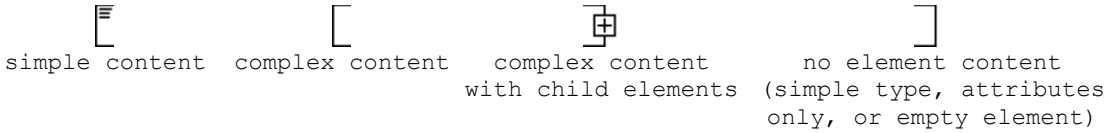
The following provides a legend for notation used in diagrams depicting XML structures.

B.1 Element Symbols

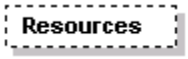
In the schema design diagrams presented above in this document, only the elements are drawn. Attributes are not visible. The cardinality of the element (0..1, 1 exactly, 0..n, 1..n) is indicated by the border of the elements. Optional elements are drawn with a dashed line, required elements with a solid line. A maximum occurrence greater one is indicated by a double border.



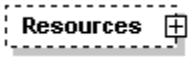
The content model of elements is symbolized on the left and right side of the element boxes. The left side indicates whether the element contains a simple type (text, numbers, dates, etc.) or a complex type (further elements). The right side of the element symbol indicates whether it contains child elements or not:



B.1.1 Examples



Optional single element without child elements. Minimum Occurrence = 0, Maximum Occurrence = 1, content = complex.



As above, but with child elements. The "plus" at the right side indicates the presence of one or more undisplayed child elements.

Mandatory single element. Minimum Occurrence = 1, Maximum Occurrence = 1, content = complex, no child elements (i.e. this denotes an *empty element*). The gray or green text below the element displays the xml-schema annotation associated with the element.

Mandatory multiple element containing child elements (content = complex). This element must occur at least once (Minimum Occurrence = 1) and may occur as often as desired (Maximum Occurrence = unbounded).

Mandatory single element containing simple content (e.g. text) or mixed complex content (e.g. text with xhtml markup). Minimum Occurrence = 1, Maximum Occurrence = 1, type = xsd:string (for example), content = simple. The three lines in the upper left corner are used for both text and numeric content.

B.2 Model Symbols ("Compositors")

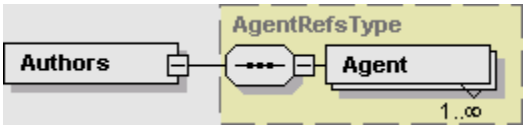
A sequence of elements. The elements must appear exactly in the sequence in which they appear in the schema diagram.

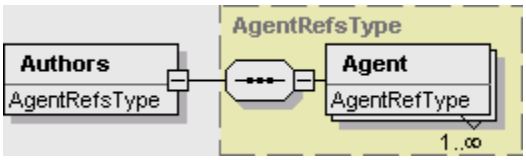
A choice of elements. Only a single element from those in the choice may appear at this position.

The "all" model, in which the sequence of elements is not fixed.

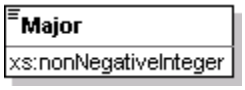
B.3 Types

If an element refers to a complex global type, the type is shown with a border.





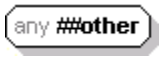
The type names of simple types are shown as well:



B.4 Model Groups and References

An *element group* is a named container with one or several elements. The group of elements can be reused at multiple places in the schema. Model groups are invisible in the instance document. Model groups have been used sparingly since they do not map to a feature in object-oriented programming languages (unless they support multiple inheritance).

Important note in reading the diagrams for model groups: If the model group symbol is drawn with simple lines (i.e. not dashed), this does not imply that the elements in the model group are required. The optionality of the group depends on the optionality of elements contained in the model group. (Model groups can be made optional, e.g. to make a model group with required elements optional in some cases, but this has not been used.)



The *"any"* group is a special kind of model group. It is a placeholder for elements not defined in the schema. The "any" element defines points where the schema can be extended. After the "Any" keyword the namespace from which the elements may come is defined, for example, "##other" specifies that the extension elements may come from any namespace, except from the current schema namespace.

Element references are indicated through a link arrow in the lower left corner. They are similar to references to model groups within a schema, but instead of refining the model group, they directly refer to a single global element. The global element can then be reused in multiple places.