**Media Manifest Product Definition**

**Contents**

1 Introduction 1

1.1 Document Organization 1

1.2 Document Notation and Conventions 1

1.3 Normative References 1

1.4 Informative References 2

1.5 XML Namespaces 3

1.6 Identifiers 3

1.7 Status 3

2 Product Definition 4

2.1 Product Definition Evolution 4

2.1.1 Offering/Avail 4

2.1.2 Fulfillment 5

2.2 Structure of Content in Media Manifest 6

3 Media Manifest 7

3.1 Media Manifest derived types 7

3.2 Identifiers 8

3.2.1 Avails ProductID and ALID 8

4 Elements of a Product Definition 10

4.1 Manifest Definition, Objects other than Inventory 10

4.2 Manifest Definition, Inventory 10

5 Mezannine File References 12

5.1 Abstract References 12

5.1.1 Common Metadata Types used for Abstract Reference 12

5.1.2 Video Reference 13

5.1.3 Audio Reference 13

5.1.4 Subtitle Reference 13

5.1.5 Image Reference 13

5.2 IMF Reference 13

5.2.1 Referencing IMF CPL Virtual Tracks 14

5.2.2 Referencing OPL Macro Outputs 15

5.2.3 Referencing an IMF Container 16

5.2.4 IMF Mapped File Set 16

**NOTE**: No effort is being made by EMA, the EMA Digital Council or Motion Picture Laboratories to in any way obligate any market participant to adhere to the Common Metadata, Common Media Manifest Metadata or Media Entertainment Core, or other specifications. Whether to adopt the specifications in whole or in part is left entirely to the individual discretion of individual market participants, using their own independent business judgment. Moreover, the EMA, the EMA Digital Council, and Motion Picture Laboratories each disclaim any warranty or representation as to the suitability of the these specifications for any purpose, and any liability for any damages or other harm you may incur as a result of subscribing to these specification.

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Description** |
| 1.0 |  | First Release |

# Introduction

MovieLabs has defined a subset of MovieLabs’ Media Manifest for use in Product Definition. This product definition is independent of the media encoding, allowing the definition to exist independent of any particular encoding.

The Manifest Product Definition is part of an overall delivery workflow that is supported by complementary specifications that can optionally be used in conjunction with this specification. Although independent of mezzanine format, specific instructions are provided for use of Interoperable Master Format (IMF, SMPTE ST-2067).



The specification and best practices for the Media Manifest can be found at [www.movielabs.com/md/manifest](http://www.movielabs.com/md/manifest). The full specification supports more complex delivery use cases, interactivity and other applications.

## Document Organization

This document is organized as follows:

1. Introduction—Background, scope and conventions
2. TBD

## Document Notation and Conventions

The document uses the conventions of Common Metadata [CM].

## Normative References

|  |  |
| --- | --- |
| [Manifest] | TR-META-MMM MovieLabs Media Manifest Metadata, v1.5, <http://www.movielabs.com/md/manifest> |
| [CM] | TR-META-CM MovieLabs Common Metadata, v2.3, <http://www.movielabs.com/md>/md  |
| [EIDR-TO] | EIDR Technical Overview, November 2010. <http://eidr.org/technology/#docs>  |
| [MEC] | Media Entertainment Core (MEC) Metadata, version 2.4. [www.movielabs.com/md/mec](http://www.movielabs.com/md/mec) and <http://entmerch.org/programsinitiatives/ema-metadata-structure>. |
| [AVAILS] | Entertainment Merchant’s Association (EMA) Avails. [www.movielabs.com/md/avails](http://www.movielabs.com/md/avails)  |
| [IMF] | SMPTE ST-2067 (family) Interoperable Master Format, [www.smpte.org](http://www.smpte.org)  |
| [SMPTE2067-2] | SMPTE ST 2067-2:2013Interoperable Master Format – Core Constraints |
| [SMPTE2067-3] | SMPTE ST 2067-3:2013Interoperable Master Format – Composition Playlist |
| [RFC3986] | IETF RFC 3986Uniform Resource Identifier (URI): Generic Syntax |
| [SMPTE 429-9] | SMPTE ST 429-9:2014D-Cinema Packaging —Asset Mapping and File Segmentation |
| [SMPTE2067-100] | SMPTE ST 2067-100:2014Interoperable Master Format – Output Profile List |
| [XML] | World Wide Web Consortium (W3C) (2004 February 4), Extensible Markup Language (XML) 1.0 (Third Edition). |
| [RFC2616] | IETF RFC 2616Hypertext Transfer Protocol — HTTP/1.1 |
| [RFC5246] | IETF RFC 5246, The Transport Layer Security (TLS) Protocol Version 1.2. |

All Common Metadata and Media Manifest references are included by reference.

## Informative References

|  |  |
| --- | --- |
| [MMDelivery] | BP-META-MMMD, Using Media Manifest, File Manifest and Avails for file Delivery (Best Practice), v1.1, [www.movielabs.com/md/manifest](http://www.movielabs.com/md/manifest) |
| [GoogleHelp] | Google Play help site. <https://support.google.com/moviestvpartners/answer/6154374>  |
| [MMC] | TR-META-MMC, Media Manifest Core, [www.movielabs.com/md/mmc](http://www.movielabs.com/md/mmc)  |

## XML Namespaces

This document defines specific use of the ‘manifest’ namespace.

‘manifest’ builds on the ‘md’ Common Metadata corresponding with Common Metadata [CM].

## Identifiers

Identifiers must be universally unique. Recommended identifier schemes may be found in Common Metadata [CM] and in DECE Content Metadata [DECEMD].

The use of Entertainment Identifier Registry identifiers ([www.eidr.org](http://www.eidr.org)) is strongly encouraged. Please see [EIDR-TO].

Best practices for identifiers can be found in Best Practices for Delivery [MMDelivery].

## Status

This specification is completed and ready for implementation. Although tested, we anticipate that additional implementation experience will yield recommendation for changes. Implementers should anticipate one or more revisions. Reasonable measures will be taken to ensure changes are backwards compatible. See Backwards Compatibility Best Practices in [CM]

# Product Definition

A studio defines a product to be sold. This product centered on a title (e.g., movie or TV episode) or collection of titles (seasons, movie series, etc.). For the purposes of an Avail, it is sufficiently defined to put it on a store site and offer it for sale. However, to deliver that title to the consumer requires a series of progressively more precise definitions, ultimately resulting in encoded bits of digital media.

As illustrated below, Sales and Fulfillment can be seen as separate processes. On the retailer side, they are linked by the Entitlement (what was sold or rented to whom). On the studios side additional information is needed to precisely define what is being availed. This specification defines a structure based on Media Manifest for communicating that information.



The Media Manifest is ideal for the progressive definitions because it is constructed as layers which can be abstract (initially) or very specific (ultimately). For example, initially a set of tracks might be defined just be language (e.g., “French subs and dubs”) and later defined as individually encoded audio and subtitle tracks packaged in a container.

## Product Definition Evolution

Product definition is a process. That is, it starts abstractly and becomes increasing more concrete as development and delivery progresses. Some of this planning occurs in pre-production and production, particularly around extended editions and bonus material; although these won’t be addressed here. This spec assumes that there is at least a list of assets that can be included in an offering.

### Offering/Avail

The first Avails can be issues months in advance of release date. At this point, the Avail is typically against a title, season or episode number. Note that it is valuable to obtain an EIDR identifier for the content because it remains constant even if attributes such as title change. As avails are tied to locales, the store language would be known, but the set of available subs and dubs might not.

At this point, the product can be sold (including pre-order), rented or made available for subscription resulting in an entitlement. The entitlement might be internal to the retailer or it might be in a common repository such as UltraViolet or Disney Movies Anywhere (DMA).

At some data, the content will be available for streaming or download.

### Fulfillment

Fulfillment requires the product to ultimately be delivered as digital data. This involves both the generation of that digital media, in what we’re calling mezzanine format, and the matching of the digital media with the product definition.

Linkage differs slightly depending on when and where this matching occurs. Early on, the linkage is abstract (e.g., “The French dub Track”). Later it becomes more specific, (“Track nnn in File xxx). These specific references tend to container and media formats. For example, if the mezzanine files are in the Interoperable Master Format (IMF) format, the linkage is to IMF UUIDs, Composition Playlists (CPLs) and/or Output Profile Lists (OPLs). If the mezzanine files are .mov files, then the linkage is typically file names and track IDs.

The following illustrates a potential evolution of product definition for a film. Note that the order is not the same for every film or every studio. These processes will also overlap.

1. Film planned for distribution
	* Title, possible working title
	* Register Abstract EIDR (title level)
2. Associated assets tentatively planned
	* This can include trailers, EPKs, various bonus material, interactive components, etc.
	* Process for these is essentially the same as the film, although they will be on their own schedules until the end
3. Product definition begins to take form
	* What subs and dubs will be necessary
	* International edits and localizations (e.g., language or culturally specific imagery). Note that some of this will depend on results from local ratings bodies.
	* Specialized video and audio tracks (HDR, WCG, HFR, object-based sound, etc.
	* Which trailers, clips, bonus material and interactivity will be included in various packages. At this point planning might be “with bonus or without bonus” without more specificity.
	* Cover art, hero art and other image components are planned (at least tentatively)
	* Initial edit-level EIDRs are registered
4. Mezzanine files are created for various media components.
	* Video, audio and subtitles can be packaged together into forms like IMF, or separate files can be generated.
	* High quality images are generated.
	* Manifestation EIDRs can be generated for these mezzanine files
5. Final deliverable to retailer is generated
	* Media could be standard mezzanines or retailer-specific formats
	* Market and retailers-specific packages are created
	* Content delivered to retailer using Media Manifest Core (MMC) [MMC] or more broadly Best Practices for Delivery [MMDelivery].

Although Media Manifest can provide useful functionality earlier, this document focuses on activities in #3, forward with particular emphasis on #4 activities.

Product Definition includes well-defined subset of Media Manifest. Structurally, Product Definition matches Media Manifest for Delivery [MMDelivery], Media Manifest Core (MMC) [MMC] and Media Manifest for Interactivity (TBS).

The primary difference between Product Definition and other uses of Media Manifest are in the Inventory section of the Media Manifest. That is because product structure is the same regardless of where it is in the supply chain, but the media assets are different. By design, these changes are isolated to the Inventory.

## Structure of Content in Media Manifest

Media Manifest [Manifest], Section 8.1 and Media Manifest Core [MMC], Section 2. MMC provides examples of how movies and TV are encoded. As the structure of the Manifest is the same at any point in delivery, these guidelines and example apply directly. The only difference in Product Definition is that there are different rules for Inventory.

# Media Manifest

## Media Manifest derived types

Media Manifest [Manifest] includes elements provides a layered model of information that abstracts various portions of content delivery. The following illustrates the subset of the Media Manifest used in the Media Manifest Core.



This specification uses the following abstractions:

* **Media** – Video, audio, subtitles, images, applications, etc. This specification defines neither the media nor its delivery. These processes should not be impacted by the use of the Manifest
* **Inventory** – The inventory describes (metadata) and points to media. I references down to individual tracks, images, etc. This provides the necessary abstraction to the media which, as mentioned above, can be in any format.
* **Grouping** – The manifest groups content that is related at playback. Presentation and Picture support are required; while Applications and Text are optional.
	+ Presentation – Tracks that play together
	+ Picture – Images that are related. This includes both metadata images (i.e., cover/key art). In other applications, this includes image galleries.
* **Experience** – Connects related content (e.g., trailers to a movie)

All mandatory elements and attributes must be included. Any optional elements may be included. The following elements and attributes are required for MMDC usage, regardless of whether they are optional, except as noted.

The following table uses the following conventions:

* Structure is given by table indentation. Parent level elements to the left.
* Attributes begin with ‘@’. For example, @ContentID refers to the ContentID attribute

## Identifiers

The identifier mechanism is very flexible, but understanding a few principles and following a few rules keeps things simple. First a reminder, IDs are formatted as follows: “md:”<type> “:”<scheme>“:”<SSID>. For example, md:experienceid:eidr-s:5EE7-A973-819A-DC1A-CDD8-H

* Opaqueness – IDs work as a string that you don’t need to examine for the Manifest to work correctly. Note that IDs are case insensitive (case should be ignored when matching)
* Uniqueness – Two IDs cannot have the same value (string). The use of <type> with values like ‘experienceid’, ‘cid’ and ‘audtrackid’ ensures that different types of IDs are not confused. The combination <scheme> and <SSID> ensured ID values are unique.
* Base ID extraction – An ID of a given <scheme> can be extracted from the Manifest ID. For example, if the scheme is “eidr-s”, the EIDR ID follows the colon. md:experienceid:eidr-s:***5EE7-A973-819A-DC1A-CDD8-H***. If eidr-x is used, the EIDR is between that colon and the next colon; for example, md:experienceid:eidr-s:***5EE7-A973-819A-DC1A-CDD8-H***:trailer.1. Other ID types can be similarly extracted.

Identifiers should use the “md:” form, preferably using EIDR. Note that in almost all cases the Edit-level EIDR is used. When there is only one offering for a given title, use the EIDR-S form. For example, md:cid:eidr-s:<edit-level EIDR>. If you can’t use EIDR, use the ‘org:’ form; for example, md:cid:org:craigsmovies.com:12345ABCDEF.

### Avails ProductID and ALID

This section addresses specifically how to populate ALIDExperienceMap/ALID when using EMA Avails [AVAILS].

If the XML version of EMA Avails is used, the /Avail/ALID is used in this field. In this case, a match is a case-insensitive string match.

If the Excel version of Avails is used, the answer is more complicated as the ‘md:’ identifier form is not typically used in the spreadsheet. The following table defines

|  |  |  |  |
| --- | --- | --- | --- |
| **ID Source** | **Spreadsheet Column (example)** | **Manifest – per spec(example)** | **Manifest – acceptable(example)** |
| Cut/Edit level EIDR | ProductID (1489-49A2-3956-4B2D-FE16-5) | EIDR-S form (md:alid:eidr-s:1489-49A2-3956-4B2D-FE16-5) | EIDR (1489-49A2-3956-4B2D-FE16-5) |
| Studio-proprietary alt cut ID | AltID(VIEW\_MT\_2006) | ORG form(md:alid:org:craigsmovies.com:VIEW\_MT\_2006) | craigsmovies.com:VIEW\_MT\_2006 |

# Elements of a Product Definition

Product definition closely follows those practices defined for delivery from a Distribution Entity to a Retailer. They differ only in the Inventory.

## Manifest Definition, Objects other than Inventory

The definition for the Manifest follows the *Using Media Manifest, File Manifest and Avails for file Delivery (Best Practice),* [Delivery] with the exception of Inventory definition.

More constrained deliveries are most clearly defined (with examples) in the EMA Media Manifest Core [MMC], again with the exception of Inventory.

## Manifest Definition, Inventory

The following table identifies the objects required in the Inventory and the detail in which they are described. This table defines both an abstract specification object as well as a mezzanine file reference (e.g., IMF reference). Note that in many cases, an abstract reference to a mezzanine file is sufficient to identify specific tracks. For example, if an IMF contains a single Presentation, reference to the “French audio” track can be unambiguous. In cases where the metadata reference is ambiguous, a more specific track reference is required.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Manifest** | **Abstract** | **Mezz. Ref** |
| Track data |  | One instance for each track | (same) |
|  | Video track | Inventory/Video | Instance with defining metadata (e.g., language if language-specific) | Required metadata plus media reference |
|  | Audio Track | Inventory/Audio | Instance with defining metadata (e.g., language, codec) | Required metadata plus media reference |
|  | Subtitle Track | Inventory/Subtitle | Instance with defining metadata (e.g., language, type) | Required metadata plus media reference |
| Metadata Image |  | Referenced from metadata by ImageID.  |  |
|  | Metadata Image | Inventor/Image | Image resolution (width and height) | Required metadata plus media reference |
| Gallery | Picture Group | Picture Group with an instance for each image | Required metadata plus media reference |
|  | Gallery Image | Inventory/Image | Image resolution (width and height) | Required metadata plus media reference |

# Mezannine File References

This section defines references to specific file types.

## Abstract References

Individual tracks in a mezzanine file can be referenced with metadata as long as there is no ambiguity. For example, if a mezzanine can have at most one audio track of a given language, audio tracks can be referenced by language. However, if two or more tracks with the same language can exist, addition metadata is required to distinguish them. We call this an *ambiguous reference*. If metadata cannot be unambiguous, then a formal reference, such as found in following sections, is required.

Whether or not ambiguity exists depends on the constraints of the use cases. This section describes metadata most useful for avoiding ambiguous references.

### Common Metadata Types used for Abstract Reference

The most common metadata objects for avoiding ambiguity across track types are the following:

* Identifier – An precise identifier (e.g., an EIDR manifestation identifier)
* Type/SubType – The type of track (e.g., ‘SDH’ for a subtitle)
* Language – The language associated with the track.

#### Identifiers

Identifiers are encoded in the TrackIdentifier element that exists within Inventory/Audio, Inventory/Video, Inventory/Subtitle, Inventory/Image and Inventory/Interactive.

Identifier is usually sufficient on its own. For example, an EIDR manifestation is defined with sufficient metadata to avoid ambiguity. EIDR manifestation guidelines provide valuable guidance for disambiguation. [Ref???]

Best Practices for Delivery [Delivery], Section 3.4 provides naming conventions for creating identifiers specific to a track. If these are adhered to by all parties, they can provide unique track references. Specifically, when the manifest file creator and the mezzanine file creator will deterministically create the same identifier. Be warned that use is limited to those cases where naming conventions are established. Also, be warned that this falls into the general category of embedding metadata in identifiers which is a risky practice.

#### Type/Subtype and Language

Type is included in Inventory/Audio, Inventory/Video, Inventory/Subtitle, Inventory/Interactive and Inventory/Ancillary. SubType is included in Inventory/Ancillary.

Language is included in Inventory/Audio, Inventory/Video Inventory/Subtitle and Inventory/Image. Inventory/Video also includes SubtitleLanguage for burned or closed subtitles, and SignedLanguage visible sign language respectively.

Type and Language can often used together are often sufficient to distinguish tracks within a track type. In most use cases, descriptions like “French SDH” subtitle track is sufficient information (i.e., Language=“fr”, Type= “SDH”).

Language is often required for localized images, but is not often contained in file metadata. A distinct means of determining image metadata is required (e.g., filename). [CHS: are there any standards such as EXIF that include language for images?]

### Video Reference

TBS

### Audio Reference

TBS

### Subtitle Reference

TBS

### Image Reference

TBS

## IMF Reference

This section defines how to reference an IMF file from a Media Manifest Inventory.

Common Manifest allows user experiences to be assembled from individual assets, e.g. a video track, audio track, etc.

The Interoperable Master Format (IMF) Framework facilitates the management and processing of multiple Compositions (airline edits, special edition …) of the same high-quality finished work (feature, episode, trailer, etc.). The playback timeline of each Composition is controlled by a Composition Playlist (CPL), which is specified in SMPTE ST 2067-3 and defines a Virtual Track for each essence kind, e.g. video, audio, etc.

An IMF Output Profile List (OPL), which is specified in SMPTE ST 2067-100, can then be used to transform (scaled, cropped, etc.) Virtual Tracks to meet the specific needs of downstream distribution channels.



Figure 1. Referencing an IMF Package from Common Manifest.

As illustrated in Figure 1, this document allows Common Manifest to reference IMF CPL Virtual Tracks and IMF OPL Macro Outputs. The IMF files can be either local or accessed through HTTP or HTTPS.

### Referencing IMF CPL Virtual Tracks

#### General

A Common Manifest Inventory Asset that references an IMF CPL Virtual Track shall conform to this Section.



Figure 2. Referencing an IMF CPL Virtual Track.

#### TrackReference

The TrackReference element is not used and should not be present.

#### TrackIdentifier

The TrackIdentifier/Namespace element shall be "smpte:imf".

The TrackIdentifier/Identifier element shall conform to the <track-id> syntax:

<track-id> ::= "cpls/" <cpl-id> "/virtual-tracks/" <virtual-track-id>

<cpl-id> ::= <uuid>

<virtual-track-id> ::= <uuid>

where <cpl-id> shall be the Id of the Composition Playlist (see Section 6.1.1 of SMPTE ST 2067-3) and <virtual-track-id> shall be equal to the TrackId of the CPL Virtual Track (see Section 6.9.3 of SMPTE ST 2067-3).

The TrackIdentifier/Location element is not used and should not be present.

#### ContainerReference

If present, the ContainerReference element shall conform to Section 3.

### Referencing OPL Macro Outputs

#### General

A Common Manifest Inventory Asset that references an OPL Macro Output shall conform to this Section.

#### TrackReference

The TrackReference element is not used and should not be present.

#### TrackIdentifier

The TrackIdentifier/Namespace element shall be "smpte:imf".

The TrackIdentifier/Identifier element shall conform to the <track-ref> syntax:

<track-ref> ::= "opls/" <opl-id> "/" <macro-output-handle>

<opl-id> ::= <uuid>

where <opl-id> is the Id of the Output Profile List and <macro-output-handle> is the Macro Instance Output Handle, as specified in SMPTE ST 2067-100.

The TrackIdentifier/Location element is not used and should not be present.

#### ContainerReference

If present, the ContainerReference element shall conform to Section 3.

### Referencing an IMF Container

#### ContainerLocation

The ContainerLocation element shall reference the Asset Map document of a Mapped File Set that conforms to Section 4, using either:

* A relative-path reference, without query or fragment component, as specified in RFC 3986. The relative-path reference shall be resolved relative to the URI of the Common Manifest document.
* An absolute-URI as specified in RFC 3986. One of the following schemes shall be used: "http" or "https".

The Mapped File Set shall include all assets necessary to process the Common Manifest Inventory Asset.

#### ContainerIdentifier

The ContainerIdentifier element is not used and should not be present.

#### ParentContainer

The ParentContainer element is not used and should not be present.

### IMF Mapped File Set

#### General

IMF assets are delivered using the Asset Map mechanism specified in SMPTE ST 429-9. The following defines a Mapped File Set (see Section 9 of SMPTE ST 429-9) for use with Common Manifest deliveries.

#### Asset Map Document Location

The Asset Map document shall be identified by an absolute-URI whose last path segment is equal to "ASSETMAP.xml".

#### Asset Map Document Encoding

The Asset Map document shall be encoded using UTF-8 as specified in W3C Extensible Markup Language 1.0

#### Asset Location and Path Constraints

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

Each Asset referenced by the Asset Map shall be identified by an absolute-URI constructed by resolving, as specified in RFC 3986, its Path element relative to the Asset Map URI.

Every location traversed by this resolved relative-path reference shall be at the same or deeper hierarchical level than the Asset Map URI.

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 128 characters, and the value of the Path element shall not exceed 1024 characters in length.

A Path element value shall have no more than 10 segments.

No two paths in an Asset Map shall have identical values, regardless of case.

#### Retrieving Resources

The resources identified by a URI using the "http" or "https" schemes shall be retrieved as the response to a GET request to the URI.