-OSTA.org-
Optical Storage
Technology Association

Digital Music, Photo, and Video Collections

MPV Interoperability Specification

Revision 1.0

25. July 2005

POINT OF CONTACT

OSTA David Bunzel OSTA President Tel: +1 (408) 253-3695 e-mail: dbunzel@osta.org http://www.osta.org/ MPV Website http://www.osta.org/mpv/	Technical Content Pieter van Zee MPV Initative Lead Editor, MPV Specification Tel: +1 541-715-8658 e-mail: Pieter.van.Zee@hp.com Felix Nemirovsky Chairman, MPV Committee Tel: +1 415 643 0944 e-mail: felix@chubaconsulting.com Young-Yoon Kim Chairman, MPV CE Working Group Tel: +82 31-200-4791 e-mail: yoyokim@samsung.com Seong-Kook Shin Editor, MPV Interoperability Specification Tel: +82 11-9615-7299 e-mail: cinsk.shin@samsung.com

ABSTRACT

The MPV Interoperability Specification tightly defines the MPV implementation practices and metadata required to guarantee interoperability of music, photo, and video playlists on consumer electronics products.

COPYRIGHT NOTICE

REVISION HISTORY

Revision	*Date*	*Comments*
1	07/25/05	Version 1.0 released.

ACKNOWLEDGEMENTS

We would like to thank many individual who dedicates their precious time and knowledge in developing this document. The unordered list of each individual goes here.

Du-Il Kim
Kenji Ichimura
Kazuyuki Shibuya
Eric Shalkey
Pieter van Zee
Hyok-Sung Choi
Lee Prewitt
Timothy Whitcher
Hee-Yeon Kim
Jin-Yong Ahn

All other members of OSTA MPV Working Group

IMPORTANT NOTICES
MPV-IS
(MPV INTEROPERABILITY SPECIFICATION)
Aug 8, 2005

(a) THIS DOCUMENT IS AN ADOPTED, PUBLISHED AND COPYRIGHTED PUBLICATION OF THE OPTICAL STORAGE TECHNOLOGY ASSOCIATION (OSTA). THE SPECIFICATION CONTAINED HEREIN IS THE EXCLUSIVE PROPERTY OF OSTA THIS DOCUMENT MAY BE COPIED IN WHOLE OR IN PART PROVIDED THAT NO REVISIONS, ALTERATIONS, OR CHANGES OF ANY KIND ARE MADE TO THE MATERIALS CONTAINED HEREIN.

(b) PRIOR TO THE RELEASE OR SALE OF ANY COMMERCIAL PRODUCT WHICH UTILIZES THE MPV INTEROPERABILITY SPECIFICATION A LICENSE MUST BE OBTAINED FROM OSTA AND THE TERMS AND CONDITIONS OF THE LICENSE MUST BE SATISFIED. INFORMATION CONCERNING THE LICENSE IS AVAILABLE AT www.osta.org/mpv/licensing.

(c) COMPLIANCE WITH THIS DOCUMENT MAY REQUIRE USE OF ONE OR MORE FEATURES COVERED BY THE PATENT RIGHTS OF AN OSTA MEMBER OR THIRD PARTY. NO POSITION IS TAKEN BY OSTA WITH RESPECT TO THE VALIDITY OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT, WHETHER OWNED BY A MEMBER OF OSTA OR OTHERWISE. OSTA HEREBY EXPRESSLY DISCLAIMS ANY LIABILITY FOR INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS OF OTHERS BY VIRTUE OF THIS OSTA DOCUMENT, NOR DOES OSTA UNDERTAKE A DUTY TO ADVISE USERS OR POTENTIAL USERS OF OSTA DOCUMENTS OF SUCH NOTICES OR ALLEGATIONS. OSTA HEREBY EXPRESSLY ADVISES ALL USERS OR POTENTIAL USERS OF THIS DOCUMENT AND SPECIFICATIONS TO INVESTIGATE AND ANALYZE ANY POTENTIAL INFRINGEMENT SITUATION, SEEK THE ADVICE OF INTELLECTUAL PROPERTY COUNSEL AND, IF INDICATED, OBTAIN A LICENSE UNDER ANY APPLICABLE INTELLECTUAL PROPERTY RIGHT OR TAKE THE NECESSARY STEPS TO AVOID INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT. OSTA EXPRESSLY DISCLAIMS ANY INTENT TO PROMOTE INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT BY VIRTUE OF THE EVOLUTION, ADOPTION, OR PUBLICATION OF THIS OSTA DOCUMENT.

(d) OSTA MAKES NO REPRESENTATION OR WARRANTY REGARDING ANY SPECIFICATION, AND ANY COMPANY USING A SPECIFICATION SHALL DO SO AT ITS SOLE RISK, INCLUDING SPECIFICALLY THE RISKS THAT A PRODUCT DEVELOPED WILL NOT BE COMPATIBLE WITH ANY OTHER PRODUCT OR THAT ANY PARTICULAR PERFORMANCE WILL NOT BE ACHIEVED. OSTA SHALL NOT BE LIABLE FOR ANY EXEMPLARY, INCIDENTAL, PROXIMATE OR CONSEQUENTIAL DAMAGES OR EXPENSES ARISING FROM THE USE OR IMPLEMENTATION OF THIS DOCUMENT. THIS DOCUMENT DEFINES ONLY ONE APPROACH TO COMPATIBILITY, AND OTHER APPROACHES MAY BE AVAILABLE IN THE INDUSTRY.

(e) THIS DOCUMENT MAY BE REVISED BY OSTA AT ANY TIME AND WITHOUT NOTICE AND USERS ARE ADVISED TO OBTAIN THE LATEST VERSION. IT IS INTENDED SOLELY AS A GUIDE FOR ORGANIZATIONS INTERESTED IN DEVELOPING PRODUCTS WHICH CAN BE COMPATIBLE WITH OTHER PRODUCTS DEVELOPED USING THIS DOCUMENT. THIS DOCUMENT AND THE SPECIFICATIONS ARE PROVIDED “AS IS”.

(f) MPV, MusicPhotoVideo AND THE MPV LOGO ARE TRADEMARKS OF OSTA. ALL OTHER TRADEMARKS ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS. THE TRADEMARKS MPV, MusicPhotoVideo AND THE MPV LOGO MAY NOT BE USED EXCEPT FOR JOURNALISTIC PURPOSES WITHOUT AN EXPLICIT LICENSE FROM OSTA. INFORMATION REGARDING THE MPV TRADEMARK LICENSE IS AVAILABLE AT www.osta.org/mpv/licensing.

(g) THE MPV SPECIFICATIONS MAY BE USED ONLY FOR PRODUCTS USED LAWFULLY. OSTA DOES NOT PROMOTE OR CONDONE THE USE OF THE SPECIFICATIONS IN ANY PRODUCT WHICH IS USED FOR AN UNLAWFUL PURPOSE.

Contents

Table of MPV Requirements

The following table is a comprehensive list of the requirements specified by the MPV Interoperability Specification, ordered by Requirement Number. This list can provide a quick reference guide to the specification and also is useful to quickly lookup more information about a specific requirement by number.

Note that the Requirement Number is not in lexicographical order. Each requirement is strongly connected with the Number, and the relationship will not be broken in further releases of this document.

WRITER REQUIREMENTS

READER REQUIREMENTS

1 Introduction

1.1 Executive Summary

MPV (MusicPhotoVideo) is an open, multi platform specification for playlists and asset management of digital music, photo, and video files. MPV makes easier the representation, exchange, processing and playback of collections of digital media content, including music, still images, stills with audio, still sequences, video clips, and audio clips.

The MPV Interoperability Specification, which is defined by this document, is a well-defined usage of MPV that provides guaranteed interoperability between playlist creator and playlist reader applications and devices.

The MPV Interoperability Specification can be used in consumer electronics products such as a CD or DVD player or on a PC. MPV playlists that conform to the Interoperability Specification and the content reference by them (like MP3 and JPEG files) are unlike audio CDs and DVD-Video discs because they store the multimedia content in data formats used by PCs.

MPV is an open format developed under the leadership of the Optical Storage Technology Association (OSTA) and available from OSTA at no cost. Information regarding the MPV Specification, SDK for software developers, and verification tools can be found at http://www.osta.org/mpv/.

Use of the MPV acronym and MusicPhotoVideo is defined in Important Notices.

A. SITUATION TODAY

Consumers create CDs full of personal digital content music, photo, and video content in PC file formats like MP3, JPEG and MPEG captured by digital cameras, photofinishing retailers, personal music collections, and PC multimedia applications. The expectation and desire of consumers is to enjoy the playback experience not only in PCs but wherever a player can go -- in their home entertainment environment, in their car or in their pocket.

Today, most consumer electronic (CE) devices do not recognize the content on CDs created by PC applications or retail services or do so poorly. Because each application stores the content on a disc uniquely, there is no standard way for CD and DVD players to recognize and playback the content. And the user playback experience is different between each CE device.

Without a standard method for organization and access, the CE device can take many minutes reading through large collections of multimedia content or will present file system views of the data. Consumers get frustrated with trying to find and quickly access their desired music, photo, or video content.

CE devices are starting to add support for PC formats. MP3 for music has enjoyed wide spread adoption in DVD player, car stereo, and personal music systems. Support for JPEG for photos is growing in consumer electronics products. MPEG1 and MPEG2 video is already broadly adopted by both PC and CE industries. Additional formats are emerging and growing in popularity, such as Windows Media Audio (WMA), ATRAC3, MPEG4 Audio (AAC), and more.

B. MPV BENEFITS CONSUMERS

When applications and CD devices both write and read MPV, consumers will enjoy enhanced interoperability of content between applications and devices from any vendor. This gives consumers more choice and vendors greater ability to innovate and differentiate.

The MPV Interoperability Specification provides interoperability of navigation and asset management. Between producers and consumers of compatible formats, the MPV Interoperability Specification provides an excellent and richly featured playlist interoperability experience.

The MPV specifications do NOT specify any required media file format; MPV is focused on successful interchange of media file collections and associated metadata, and can be applied to any kind of media file format. Other specifications may choose to leverage MPV for media collection and playlist interchange, while additionally specifying specific media file format and storage media encodings.

While MPV does not provide specifications for interoperable media file formats and thus cannot guarantee playback of the media file themselves, devices conforming to the MPV Interoperability Specification provides a valuable experience of another kind: the user will see a list of all referenced content on a storage media, and can navigate it all, even if it cannot play all items. Items that cannot be played can remain visible, providing the user with a consistent user experience across all MPV-enabled devices and applications, even if the user is unable to play some of the content items.

For consumers burning CDs on PCs, applications supporting MPV will create MPV CDs that can be viewed in other PCs and provide the additional benefit of playing back in CE devices such as CD or DVD players from many manufacturers. For consumers sending film or digital content to a retail photo service for storage on a photo CD, MPV CDs will provide an archive of the high resolution content as well as a consistent playback experience in DVD players and TV screens.

If digital cameras implement MPV, consumers wishing to enjoy the playback experience in their TV or DVD system will benefit from MPV organization to maintain the user playback experience when the memory card is removed from the camera. TVs or DVDs supporting MPV will playback the content from any camera in a consistent manner.

1.2 MPV Interoperability Specification Guaranteed Navigation, Enhanced Playback Experience

The MPV Interoperability Specification provides a standard way to organize and navigate content and to play it back. Think of the MPV Interoperability Specification compliant control file (Manifest) as a table of contents or index on the disc or flash memory card that a CE device with a MPV reader can recognize and play. The control file defines the contents of the storage media and enables the CE device to quickly find, access and present the consumer s multimedia content. MPV Interoperability Specification-enabled playback devices quickly recognize the control file and present the user with a playlist and simple navigation of the content. The MPV Interoperability Specification allows the user or creative application to organize the media files any way they wish on the storage media using long user-friendly filenames or titles. Additions and changes to playlists can be made easily and quickly without rewriting the entire storage media.

In addition to providing basic playlist capability of a sequence of files, the MPV Interoperability Specification can enhance the user experience in significant ways, such as:

- Users can navigate large collections of hundreds or thousands of files organized into multiple playlists, such as songs by genre, by artist, and by album.
- Creation and playback of multimedia slide shows of pictures and video with background music and transition effects.
- Songs can have album art, lyrics, music in both audio-only and video file formats, and multiple encodings of the same music, for example WMA and MP3.
- MPV manages multiple renditions of images, such as high resolution, screen resolution, and thumbnails that enable high-performance playback on low-end systems using low-res images and printing using high-res images.

This document establishes a fixed usage of MPV that is used to guarantee interoperability between playlist creator and playlist reader.

However, it does not require any specific set of media file formats. Thus, it is quite possible to have a storage media filled with media files and MPV playlists that a MPV-enabled player can navigate but cannot

play because it cannot read the media file formats. While this can be a disappointing user experience, there is still tremendous value in establishing an interoperable playlists format because the playlist format can be stable while the media file formats continue to evolve.

1.3 No Content Playback Guarantee

The MPV Interoperability Specification does NOT establish a required set of media file content formats that must be produced by writer applications and must be played by reader applications and devices. While the MPV Interoperability Specification ensures that a MPV player can properly read the MPV playlist information and present its contents to the user, it does NOT ensure that the content itself can be played by the MPV player.

To provide guarantees of content playback, the MPV Interoperability Specification can be combined with a well-defined set of content file formats. The ability to do this is available to any entity or organization that wants to do so. In a typical content type specification, the specification would reference the MPV Interoperability Specification and provide a well-defined set of content formats to be used in combination.

The content formats would be of types relevant to the entity or organization. For example, an audio books standard might specify only audio formats, while a digital camera standard might specify audio, photo, and video-clip formats.

1.4 MPV Profiles

The MPV specification is developed in a modular manner and in phases. This results in "profiles". Each profile in MPV defines only those formats and practices that are necessary for the key tasks targeted by that profile. The MPV Interoperability Specification defines interoperability requirements for each of a growing number of profiles[1], including:

- Basic Profile: key tasks: defining content collections, renditions, identifiers, and access to other metadata
- Presentation Profile: two key tasks: viewing a slide show and interactively browsing content collections
- Music Profile: key tasks: listening to a music collection and interactively browsing content collections

Underlying all profiles is the Core, which defines the overall framework of all MPV profiles. The Basic and Presentation Profiles, for example, build on the Core and, when implemented in consumer electronics devices like DVD players or in application software, can provide compelling playback of photo-video slide shows and interactive browsing of photo-video content. It can also facilitate interchange of photo-video content between applications. The Presentation Profile is also used by the Music Profile too as a music playlist.

1 Each new MPV profile introduced into MPV specification shall have separate chapters.

2 MPV Interoperability Specification 1.0

2.1 MPV Interoperability Specification Introduction

The MPV Interoperability Specification makes use of existing MPV specifications ([MPVCore], [MPVBasic], [MPVPres], [MPVMusic]) and combines them with additional specific requirements to define tightly the usage of these MPV profiles to guarantee interoperability between devices and applications that conform to the MPV Interoperability Specification.

The MPV Interoperability Specification introduces no new schema or metadata. The entire MPV Interoperability Specification consists of practices that dictate how the other MPV specifications are used. The objective of the MPV Interoperability Specification is to make consistent both MPV playlist writing, reading, and playback. Notably, this is achieved in some cases by overriding or limiting the use of certain aspects of the referenced MPV specifications, where their capabilities are too broad, to imprecise, or otherwise unsuitable for broad implementation across a full range of applications and devices, including those with limited capacities, such as limited RAM and Flash ROM memory and CPU and I/O performance.

These limitations and consistent practices enable writers and readers to be highly interoperable, enabling consumers to expect and experience compatibility and interoperability of MPV devices.

2.2 MPV Interoperability Specification Practices

The MPV Interoperability Specification establishes three important practices.

1. All practices are qualified using the keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, if and where they appear in this document, are to be interpreted as described in [RFC2119].

   The keywords are classified into three imperative levels. All words at a given level have the same level of imperative.

   - Level 1: MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT
   - Level 2: SHOULD, SHOULD NOT
   - Level 3: RECOMMENDED, MAY, OPTIONAL

   For conformance testing, the keyword imperative levels are treated as warning levels, with the following meaning:

   - Level 1: Error -- MUST be fixed.
   - Level 2: Severe Warning -- SHOULD be fixed to enhance compatibility.
   - Level 3: Warning RECOMMEND to be fixed. Not critical to compatibility.

2. All practices are classified as Writer, or Reader requirements. Writer requirements identify specific MPV content that Writers must produce; Reader requirements identify specific behaviors that Readers must implement.

3. All practices are identified with a unique requirement number by which they may be referenced easily.

   For example, a verification tool could output results and reference the requirement by number. Practices are identified using a prefix plus a number. This specification uses the letters PL plus the letter W for Writer, or R for Reader, plus the specification version number without the decimal separator as the prefix, as in PLW100-83.

2.3 Formalities For Use of the MPV Interoperability Specification

The mechanism that MPV uses to define extensions to the Core specification is the Profile. MPV Core sets out specific formalities to follow when a MPV Profile is used -- a MPV file must declare which profiles it implements and it must declare the namespaces of the profiles. This allows a processing application to quickly determine whether a given MPV file meets its expectations for processing.

2.3.1 SCHEMA NAMESPACE

The MPV Interoperability Specification does not define any schema. Therefore no schema namespace is defined. However, profile namespaces are defined[2]

2.3.2 COMPATIBILITY

The MPV Interoperability Specification 1.0 is fully compatible with the MPV framework established by [MPVCore]. Thus MPV files that implement the MPV Interoperability Specification 1.0 should be usable in basic ways by most MPV-aware applications and devices that are not in conformance with the MPV Interoperability Specification.

Applications and devices that conform to the MPV Interoperability Specification 1.0 will provide the user a consistent user experience.

2.3.3 INTEROPERABILITY PROFILES

The MPV Interoperability Specification 1.0 defines interoperability requirements for existing MPV specifications that are more general purpose and have more degrees of flexibility than can be accommodated for interoperability with all manner of devices and applications. These specifications are:

- MPV Basic Profile 1.0 Specification
- MPV Presentation Profile 1.0 Specification
- MPV Music Profile 1.0 Specification

For each of these specifications, the MPV interoperability Specification defines an additional level of compliance. For each profile, an additional MPV Interoperability Profile namespace is defined, which MUST be used within the <file:ManifestProperties> element in the Manifest to indicate that a given MPV manifest complies with an Interoperability profile.

In addition, these underlying specifications are utilized[3]:

- XML Manifest 2.0 Specification
- MPV Core 1.0 Specification

The MPV Interoperability Specification 1.0 includes the schema and practices detailed by this document.

2.4 Terminology

Some terms must be introduced before reading this document. This document is known as "MPV Interoperability Specification". Additionally "MPV IS" or "IS" is used for the abbreviation. Note not all

2 Note that MPV IS does define new <file:Profile> namespaces, see Section 3.1.2, 4.1.2, and 5.1.2 for details.

3 Note that these are not the full list of referring documents. See the appendix.

specific terms are covered here. Such terms are covered in other specifications. See the appendix for exact information.

2.4.1 Reader, Writer, and Applications.

In this document, a MPV Reader refers to an application or a software module that discover, reads, and understands a MPV manifest. The resulting information is used to give users various experiences.

A MPV Writer refers to an application or a software module which generates a MPV manifest.

An application is a device or software program that implements either or both a MPV Reader and MPV Writer. A MPV Writer does not have to be a Reader, nor does a Reader have to be a Writer. In practice, many Readers are also Writers.

2.4.2 Top-level Asset

An asset is called a top-level asset if and only if the asset is not referenced by any other asset.

For example, in the next example MPV document, we can see two top-level assets; one <mpv:Still> asset and one <mpv:Audio> asset.

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest ...>
  ...
  <mpv:AssetList>
    <mpv:Still mpv:id="still001">
      ...
    </mpv:Still>
    <mpv:Audio mpv:id="audio001">
      ...
    </mpv:Audio>
  </mpv:AssetList>
</file:Manifest>
```

In the following example, we have only one top-level asset, <mpv:Seq>.

```
<file:Manifest ...>
  ...
  <mpv:AssetList>
    <mpv:Still mpv:id="still001">
      ...
    </mpv:Still>
    <mpv:Audio mpv:id="audio001">
      ...
    </mpv:Audio>
    <mpv:Seq mpv:id="seq001">
      <mpv:StillRef mpv:idRef="still001"/>
      <mpv:AudioRef mpv:idRef="audio001"/>
    </mpv:Seq>
  </mpv:AssetList>
</file:Manifest>
```

2.4.3 Child asset

A child asset is an asset that is referenced by other composite assets. In other words, any asset referenced by a composite asset is a child asset. Thus we can safely say that <mpv:AssetList> consist of top-level assets and child assets.

In the first example of the previous section, we have no child assets, whereas we have two child assets in the second example of the previous section.

3 Basic Profile Interoperability Specification

This chapter covers basic requirements and guidelines for all MPV-aware applications and devices. The main concerns here are strongly related to the [MPVCore], [MANIFEST], [DC-NMF], and [MPVBasic]. Additional profiles such as MPV Presentation Profile, MPV Music Profile, etc are dealt with in following chapters.

Applications that implement the IS-specific MPV Basic Profile MUST use the proper namespace as discussed in Section 3.1.2.

3.1 Writer Requirements & Guidelines

We will discuss requirements and guidelines for all MPV Writers in this section. Applications which generate MPV manifests should follow all the requirements and guidelines in this section.

3.1.1 Support MPV Basic Profile

PLW100-1 MPV applications MUST support the MPV Basic Profile Specification.

The MPV Basic Profile Specification describes a baseline usage of MPV and is the common underlying level of MPV interoperability. MPV Basic Profile is the only profile that is mandatory for all applications complying with the MPV Interoperability Specifications.

According to [MPVCore] Section 2.1, MPV Core Specification cannot be used itself, so the MPV Basic Profile is used to provides a basic manifest and <mpv:AssetList> element. Thus, MPV applications MUST support at least MPV Basic Profile.

See also Section 3.1.2, 3.1.5.

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest
       xmlns:file="http://ns.osta.org/manifest/1.0/"
       xmlns:file2="http://ns.osta.org/manifest/2.0"
       xmlns:mpv="http://ns.osta.org/mpv/1.0/"
       xmlns:mpvp="http://ns.osta.org/mpv/presentation/1.0/"
       xmlns:mpvm="http://ns.osta.org/mpv/music/1.0/"
       xmlns:nmf="http://ns.osta.org/nmf/1.0/"
       ...
  >
  <nmf:Metadata>
    <file:ManifestProperties>
      <file:ProfileBag>
        <file:Profile>http://ns.osta.org/mpv/basic/1.0/</file:Profile>
        ...
      </file:ProfileBag>
      <file2:AboutManifestMPVDocumentID>DOC001</file2:AboutManifestMPVDocumentID>
      <file2:WrittenBy>http://www.companyA.com/Device/Model1934/</file2:WrittenBy>
    </file:ManifestProperties>
  </nmf:Metadata>
  ...
  <mpv:AssetList>
    ...
    <mpv:Document mpv:id="DOC001">
      ...
```

```
    </mpv:Document>
    ...
  </mpv:AssetList>
</file:Manifest>
```

Reference: Section 2.1 of [MPVCore], Section 2.3 of [MPVCore], [MPVBasic]

3.1.2 Basic Profile IS Profile Identifier

PLW100-2 Declare IS-Compliant Profile. A MPV Manifest that complies with the requirements of the MPV Basic Profile Interoperability Specifications (this section) MUST declare that compliance by listing the Basic Profile IS Profile identifier in a <file:Profile> entry in the <file:ManifestProperties>.

The MPV Basic Profile Interoperability Specification profile identifier is:

Interoperability Specification For ...	*IS Profile Identifier*
Basic Profile	http://ns.osta.org/mpv/basic/is/1.0/

Use of this IS profile identifier allows a MPV Reader to determine that the manifest complies with these Basic Profile Interoperability Specifications.

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest
       xmlns:file="http://ns.osta.org/manifest/1.0/"
       xmlns:file2="http://ns.osta.org/manifest/2.0"
       xmlns:mpv="http://ns.osta.org/mpv/1.0/"
       xmlns:mpvp="http://ns.osta.org/mpv/presentation/1.0/"
       xmlns:mpvm="http://ns.osta.org/mpv/music/1.0/"
       xmlns:nmf="http://ns.osta.org/nmf/1.0/">
  <nmf:Metadata>
    <file:ManifestProperties>
      <file:ProfileBag>
        <file:Profile>http://ns.osta.org/mpv/basic/1.0/</file:Profile>
        <file:Profile>http://ns.osta.org/mpv/basic/is/1.0/</file:Profile>
        ...
      </file:ProfileBag>
      <file2:AboutManifestMPVDocumentID>DOC001</file2:AboutManifestMPVDocumentID>
      <file2:WrittenBy>http://www.companyA.com/Device/Model1934/</file2:WrittenBy>
    </file:ManifestProperties>
  </nmf:Metadata>
  ...
  <mpv:AssetList>
    ...
    <mpv:Document mpv:id="DOC001">
      ...
    </mpv:Document>
    ...
  </mpv:AssetList>
</file:Manifest>
```

See also Section 3.1.1, 3.1.5, 3.2.2, 3.3.2.

3.1.3 Asset Identification

PLW100-3 An asset MUST have the "mpv:id" attribute in every case.

This practice guarantees that a "mpv:id" attribute is used as the primary source of asset identification. When a MPV document contains no composite assets (such as <mpv:StillWithAudio>, <mpv:StillPanoramaSequence>, <mpv:Seq>, and so on), it is possible to generate a MPV document whose assets have no "mpv:id" attributes. However, providing "mpv:id" attributes consistently allows the Reader to work efficiently.

Note that the grammar of a "mpv:id" attribute requires that it MUST begin with an alphabetic character, as described in [MPVCore] section 5.4.

Example:

```
<mpv:AssetList>
  <mpv:Still mpv:id="still001">
    ...
  </mpv:Still>

  <mpv:Video mpv:id="video001">
    ...
  </mpv:Video>
</mpv:AssetList>
```

Reference: Section 5.4 of [MPVCore].

3.1.4 Provide only one <mpv:AssetList>

PLW100-4 MPV applications MUST provide exactly one <mpv:AssetList>.

This practice guarantees that a given MPV manifest is strictly confirming [MPVCore] usage.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest ...>
  ...
  <mpv:AssetList>
    ...
    <mpv:Document mpv:id="DOC001">
      ...
    </mpv:Document>
    ...
    <mpv:Still mpv:id="STILL001">
      ...
    </mpv:Still>
  </mpv:AssetList>
</file:Manifest>
```

Reference: Section 2.3.2 of [MPVCore].

3.1.5 Specify Correct Profile Elements

PLW100-5 One or more appropriate <file:Profile> elements MUST be used to represent proper MPV profiles.

This practice ensures that a Reader can quickly determine the usage of MPV Profiles. Although it is possible to determine the usage of MPV Profiles by analyzing the MPV document, it is preferable to generate proper <file:Profile> elements to help MPV readers.

If the document supports more than one MPV Profile, a Writer MUST provide all required <file:Profile> elements. The order of each <file:Profile> element is not important.

For example, if the document supports MPV Basic Profile, it MUST provide following <file:Profile>

element:

```
<file:Manifest ...>
  <nmf:Metadata>
    <file:ManifestProperties>
      <ProfileBag xmlns="http://ns.osta.org/manifest/1.0/">
        <Profile>http://ns.osta.org/mpv/basic/is/1.0/</Profile>
        ...
      </ProfileBag>
      ...
    </file:ManifestProperties>
  </nmf:Metadata>
  ...
</file:Manifest>
```

If the document supports the MPV Basic Profile plus the MPV Music Profile, it MUST provide the following <file:Profile> elements:

```
<file:Manifest ...>
  <nmf:Metadata>
    <file:ManifestProperties>
      <ProfileBag xmlns="http://ns.osta.org/manifest/1.0/">
        <Profile>http://ns.osta.org/mpv/basic/is/1.0/</Profile>
        <Profile>http://ns.osta.org/mpv/music/is/1.0/</Profile>
        ...
      </ProfileBag>
    </file:ManifestProperties>
    ...
  </nmf:Metadata>
  ...
</file:Manifest>
```

If the document supports the MPV Basic Profile as well as the MPV Presentation Profile, it MUST provide the following <file:Profile> elements:

```
<file:Manifest ...>
  <nmf:Metadata>
    <file:ManifestProperties>
      <ProfileBag xmlns="http://ns.osta.org/manifest/1.0/">
        <Profile>http://ns.osta.org/mpv/basic/is/1.0/</Profile>
        <Profile>http://ns.osta.org/mpv/presentation/is/1.0/</Profile>
        ...
      </ProfileBag>
    </file:ManifestProperties>
    ...
  </nmf:Metadata>
  ...
</file:Manifest>
```

If the document supports both the MPV Basic Profile and the MPV Presentation Profile plus the MPV Music Profile, it MUST provide the following <file:Profile> elements:

```
<file:Manifest ...>
  <nmf:Metadata>
    <file:ManifestProperties>
      <ProfileBag xmlns="http://ns.osta.org/manifest/1.0/">
        <Profile>http://ns.osta.org/mpv/basic/is/1.0/</Profile>
        <Profile>http://ns.osta.org/mpv/presentation/is/1.0/</Profile>
        <Profile>http://ns.osta.org/mpv/music/is/1.0/</Profile>
```

```
        ...
      </ProfileBag>
    </file:ManifestProperties>
    ...
  </nmf:Metadata>
  ...
</file:Manifest>
```

See also Section 3.1.1.

3.1.6 DC plain-text elements: creator, description, identifier, publisher, rights, title

PLW100-6 DC plain-text elements; creator, description, identifier, publisher, rights, title. When information corresponding to any or all of this basic set of defined Dublin Core elements is known, the appropriate elements SHOULD be used to represent the information. If present, an element's value MUST be UTF8-encoded plain text and has no specific interpretation other than as human-readable plain text that may be displayed or searched.

This practice is to guarantee that a Reader can read basic Dublin Core information regardless of its ability to read the information from the file format. Also, by caching the value in the MPV manifest, it protects that value from loss if the asset itself should be edited. The element value is arbitrary with no specified interpretation other than as a human-readable plain-text string suitable for display or search.

Many applications may also have formatted versions of this content. MPV does not define the interchange of formatted content. The recommended approach is to store the formatted content using metadata extensions while recording the interchangeable form of the information as DC metadata.

Example:

```
<mpv:Still mpv:id="ID000103">
  <mpv:ContentID>urn:osta-
org:mpv:dsig:md5:all:EF886AEFA3B340da971BAF09B17DBC12</mpv:ContentID>
  <mpv:LastURL>IMG001.JPG</mpv:LastURL>
  <nmf:Metadata>
  <!-- DC provides baseline interchange of metadata. -->
    <dc:Properties>
      <dc:description>Formatted content that may have any kind of rich text and even
be spread across multiple metadata elements will be provided as plain-text in the DC
element.</dc:description>
      <dc:format>image/jpeg</dc:format>
      <dc:title>Title line 1 Title line 2 Title line 3</dc:title>
    </dc:Properties>
  </nmf:Metadata>
  <mpv:Metadata>
    <!-- the app defines its own metadata schema to hold formatted content. -->
    <app:Properties>
      <app:description><p><i>Formatted</i> content that may have <b>any</b> kind of
rich text and even be spread across multiple <a
href="glossary.htm#metadata">metadata elements</a> will be provided as <i>plain-
text</i> in the DC element.</p></app:description>
      <app:title><font size=16><p><b>Title line 1</b></p></font><font
size=14><p><i>Title line 2</i></p></font><font size=12><p>Title line
3</p></font></app:title>
    </app:Properties>
  </mpv:Metadata>
</mpv:Still>
```

Confirming: [DC-NMF]

3.1.7 <dc:format> element

PLW100-7 <dc:format> element. Each asset that has at least one <mpv:LastURL> element SHOULD provide the file format of the asset's file content using the <dc:format> element as recommended by [MPVCore].

This practice is to guarantee that a Reader can quickly determine whether it can handle the asset file format without opening up the asset file. On devices with slow I/O, this is an important performance enhancement.

[MPVCore] provides a set of well-defined MIME type values for the <dc:format> element in the [MPVCore] specification. This specification may be updated over time to add additional values as needed.

Example:

```
<mpv:Still mpv:id="ID000103">
  <mpv:ContentID>urn:osta-
org:mpv:dsig:md5:all:EF886AEFA3B340da971BAF09B17DBC12</mpv:ContentID>
  <mpv:LastURL>IMG001.JPG</mpv:LastURL>
  <nmf:Metadata>
    <dc:Properties>
      <dc:format>image/jpeg</dc:format>
    </dc:Properties>
  </nmf:Metadata>
  <mpv:Rendition>
    <mpv:StillRef mpv:idRef="ID000103T"/>
  </mpv:Rendition>
</mpv:Still>
<mpv:Still mpv:id="ID000103T">
  <mpv:ContentID>urn:osta-
org:mpv:dsig:md5:all:EF886AEFA3B340da971BAF09B17DBC12</mpv:ContentID>
  <mpv:LastURL>IMG001.JPG#vnd.osta-org.exif-thumb</mpv:LastURL>
  <nmf:Metadata>
    <dc:Properties>
      <dc:format>image/vnd.osta-org.exif-thumb</dc:format>
    </dc:Properties>
  </nmf:Metadata>
</mpv:Still>
```

Confirming: [DC-NMF]

3.1.8 <dc:language> element

PLW100-8 <dc:language> element. The <dc:language> element SHOULD be used when the asset language is known. If present, the <dc:language> element MUST be interpreted as a locale as recommended in [DCNMF].

This practice is to guarantee that a Reader can determine the locale of the information regardless of its ability to read the information from the file format. The format of the <dc:language> element is a two-letter Language Code, or a two-letter Language Code, followed by a "-" and followed by a two-letter Country Code.

The <dc:language> element in the asset is unrelated to the language of the metadata. To specify the language of the metadata you MUST use a <dc:language> element in the <mpv:Document> element of the Manifest, as defined by [MANIFEST]. This means that the metadata in a MPV Manifest belongs to a single language and locale for presentation and sorting purposes, although the UTF-8 encoding of the content allows for characters from many languages to be used unambiguously.

Note that both the Language Code and the Country Code are not case-sensitive. However, it is recommended to use lowercase letters for the Language Code, and to use uppercase letters for the Country

Code.

Example:

```
<dc:Properties>
  <dc:language>en-US</dc:language>
  ...
</dc:Properties>
```

Example:

```
<dc:Properties>
  ...
  <dc:language>ko</dc:language>
</dc:Properties>
```

See also Section 3.1.17.

Confirming: [DC-NMF]

3.1.9 <dc:subject> element

PLW100-9 <dc:subject> element. The <dc:subject> element SHOULD be used when keywords about the asset subject are known. If present, <dc:subject> MUST be interpreted as one or more UTF8-encoded plain text keywords or key phrases that use the semicolon (;) without spaces as a separator. The semicolon may be used in a keyword or key phrase by escaping it with a backslash (\;). A backslash may be used by escaping it with a backslash (\\). This interpretation is a refinement of the usage defined in [MPVCore].

This practice is to guarantee that a Reader can read keyword information regardless of its ability to read the information from the file format. Also, by caching the value in the MPV manifest, it protects that value from loss if the asset itself should be edited. Each keyword or phrase is an arbitrary value with no specified interpretation other than as a human-readable plain-text string.

Example:

```
<mpv:Still>
  <nmf:Metadata>
    <dc:Properties>
      <dc:subject>Amusement;Theme;Summer</dc:subject>
    </dc:Properties>
  </nmf:Metadata>
  ...
</mpv:Still>
```

Example:

```
<mpv:Still>
  <nmf:Metadata>
    <dc:Properties>
      <dc:subject>Pets\; Kitty and Doggy</dc:subject>
    </dc:Properties>
  </nmf:Metadata>
  ...
</mpv:Still>
```

Confirming: [DC-NMF]

3.1.10 <dcterms:created> and <dcterms:modified> elements

PLW100-10 <dcterms:created> and modified elements. The <dcterms:created> and <dcterms:modified> elements SHOULD be used when these dates are known. If present, the value MUST be encoded as specified in [MPVCore].

This practice is to allow a Reader to display date information without having to access the file system. For Internet usage, this is especially valuable.

```
<mpv:Still mpv:id="still001">
  <nmf:Metadata>
    <dc:Properties>
      <dc:title>F. Valley</dc:title>
      <dc:creator>Seong-Kook Shin</dc:creator>
      <dc:description>H.Y. and I in the F. Valley.</dc:description>
    </dc:Properties>
    <dcterms:Properties>
      <dcterms:created>2004-03-14</dcterms:created>
    </dcterms:Properties>
  </nmf:Metadata>
  <mpv:LastURL mpv:filesystem="FAT32">images/still01.jpg</mpv:LastURL>
  ...
</mpv:Still>
```

The <dc:date> element is not to be used, the application MUST use the derived types such as <dcterms:created> or <dcterms:modified>.

Confirming: [DC-NMF], Section 10.5 of [MPVCore]

3.1.11 <dc:terms:extent> element

PLW100-11 <dcterms:extent> element. If the <dcterms:extent> element is used, it MUST be used to specify the size of the asset, in bytes, in an integer base 10 value without any further grammar. The size is the external size of the asset, as stored or streamed.

This practice is to allow a Reader to display size information without having to access the file system. For Internet usage, this is especially valuable.

```
<mpv:Still mpv:id="still001">
  <nmf:Metadata>
    <dc:Properties>
      <dc:title>F. Valley</dc:title>
    </dc:Properties>
    <dcterms:Properties>
      <dcterms:extent>234153</dcterms:extent>
    </dcterms:Properties>
  </nmf:Metadata>
  <mpv:LastURL mpv:filesystem="FAT32">images/still01.jpg</mpv:LastURL>
  ...
</mpv:Still>
```

3.1.12 DC metadata use

PLW100-12 DC metadata use. If a MPV Writer has Dublin Core-type information, it SHOULD write this DC content in the MPV manifest. DC values MUST take precedence over any equivalent values embedded in the asset itself.

This practice is to encourage a Writer to write Dublin Core information so that the playback experience will be more robust, allowing a Reader that cannot read an asset's file format to at least display information

about the asset.

If a Writer knows DC information at the moment that a MPV manifest is written, it SHOULD write the information to the manifest. Use of the following essential Dublin Core metadata is encouraged.

DC Elements

- creator
- description
- format
- identifier
- language
- publisher
- rights
- subject
- title

DC Terms

- created
- extent
- modified

If any of this information is available at writing time, a Writer MUST write them in Dublin Core metadata syntax. The Writer MUST NOT write this information in custom metadata format only.

Confirming: [DC-NMF]

3.1.13 Preserve Singleton attribute on DC metadata

PLW100-13 An asset MUST have only a single occurrence of a given DC element, and only the singleton plain-text elements MUST be used.

This practice guarantees fast access to the specific DC metadata information.

Bad Example:

```
<mpv:Still>
  <nmf:Metadata>
    <dc:Properties>
      <dc:title>My family</dc:title>
      <dc:title>In the South Park</dc:title>
    </dc:Properties>
    <dc:Properties>
      <dc:creator>John Smith</dc:creator>
    </dc:Properties>
  </nmf:Metadata>
  ...
</mpv:Still>
```

In the previous example, the <dc:title> element appears twice, and the <dc:creator> element is placed in a separate <dc:Properties> element. This bad practice makes it difficult for a Reader to retrieve required information in a reasonable time and correctly.

Good Example:

```
<mpv:Still>
  <nmf:Metadata>
    <dc:Properties>
      <dc:title>My family, South Park</dc:title>
      <dc:creator>John Smith</dc:creator>
    </dc:Properties>
  </nmf:Metadata>
  ...
</mpv:Still>
```

Confirming: [DC-NMF]

3.1.14 Image Size metadata

PLW100-14 <mpv:Still> Image size metadata. Every still image SHOULD have the image size metadata.

This practice guarantees that the Reader can determine the still image size without opening the media file. This is important on low-performance systems and also when MPV is used over an Internet connection with a web browser. In particular, the web browser environment does not allow direct access to the image file itself to recover size information, but size information may be critical to the correct operation of the web page or web server functionality. For example, being able to recommend that low-resolution still images not be printed larger than a certain size. The metadata to use for image size is the JPEG2000-based schema as defined by example in [MPVCore].

Example:

```
<mpv:Still>
  <mpv:ContentID>urn:osta-org:
mpv:dsig:md5:all:EF886AEFA3B340da971BAF09B17DBC12</mpv:ContentID>
  <mpv:LastURL mpv:filesystem="ISO9660-1">IMG0003.JPG</mpv:LastURL>
  <mpv:Metadata>
    <BASIC_IMAGE_PARAM xmlns="http://www.jpeg.org/jpx/">
      <BASIC_IMAGE_INFO>
        <IMAGE_SIZE>
          <WIDTH>1600</WIDTH>
          <HEIGHT>1200</HEIGHT>
        </IMAGE_SIZE>
      </BASIC_IMAGE_INFO>
    </BASIC_IMAGE_PARAM>
  </mpv:Metadata>
</mpv:Still>
```

Related: [MPVCore], [JFIF]

3.1.15 <file2:WrittenBy> element

PLW100-15 <file2:WrittenBy> element. The <file:ManifestProperties> element MUST have the <file2:WrittenBy> element, which identifies the application that wrote this specific MPV manifest. The element is updated every time the manifest is modified in any way.

This practice conforms to [MANIFEST] and is to guarantee that a Reader can quickly determine whether it knows the application that wrote the manifest. Because this attribute is in the opening element of the manifest, it enables easy transfer of the manifest processing to one of several possible readers that provide differing behaviors and levels of validation. This attribute can also be used by a MPV launcher to preferentially launch the application that wrote the manifest. Note that the value of the attribute MUST be a URN-style value to minimize the risk of value collision.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest
       xmlns:file="http://ns.osta.org/manifest/1.0/"
       xmlns:file2="http://ns.osta.org/manifest/2.0"
       xmlns:mpv="http://ns.osta.org/mpv/1.0/"
       xmlns:mpvp="http://ns.osta.org/mpv/presentation/1.0/"
       xmlns:mpvm="http://ns.osta.org/mpv/music/1.0/"
       xmlns:nmf="http://ns.osta.org/nmf/1.0/">
  <nmf:Metadata>
    <file:ManifestProperties>
      <file2:AboutManifestMPVDocumentID>DOC001</file2:AboutManifestMPVDocumentID>
      <file:ProfileBag>
        ...
      </file:ProfileBag>
      <file2:WrittenBy>http://www.companyA.com/Device/Model1934/</file2:WrittenBy>
    </file:ManifestProperties>
  </nmf:Metadata>
  ...
  <mpv:AssetList>
    ...
    <mpv:Document mpv:id="DOC001">
      ...
    </mpv:Document>
    ...
  </mpv:AssetList>
</file:Manifest>
```

Related: [MANIFEST]

3.1.16 <file2:AboutManifestMPVDocumentID> element

PLW100-16 <file2:AboutManifestMPVDocumentID> element. The <file:ManifestProperties> element MUST have the <file2:AboutManifestMPVDocumentID> element which MUST point to a proper <mpv:Document> asset.

This practice conforms to [MANIFEST] and is very useful for applications that are looking for the rich metadata information about the current document. For example, a typical use will be to look up the <file2:AboutManifestMPVDocumentID> asset, then look at its <mpv:Document> asset or its <mpv:Related mpv:relationship="derivedFrom"> element to locate a previous manifest from which the current manifest was derived.

This requirement also implies that a MPV manifest MUST have <mpv:Document> asset.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest
       xmlns:file="http://ns.osta.org/manifest/1.0/"
       xmlns:file2="http://ns.osta.org/manifest/2.0"
       xmlns:mpv="http://ns.osta.org/mpv/1.0/"
       xmlns:mpvp="http://ns.osta.org/mpv/presentation/1.0/"
       xmlns:mpvm="http://ns.osta.org/mpv/music/1.0/"
       xmlns:nmf="http://ns.osta.org/nmf/1.0/">
  <nmf:Metadata>
    <file:ManifestProperties>
      <file:ProfileBag>
        ...
      </file:ProfileBag>
```

```
      <file2:AboutManifestMPVDocumentID>DOC002</file2:AboutManifestMPVDocumentID>
      <file2:WrittenBy>http://www.companyA.com/Device/Model1934/</file2:WrittenBy>
    </file:ManifestProperties>
  </nmf:Metadata>
  ...
  <mpv:AssetList>
    <mpv:Document mpv:id="DOC001">
      <mpv:LastURL>metahist/hist0001.pvm</mpv:LastURL>
      ...
    </mpv:Document>
    ...
    <mpv:Document mpv:id="DOC002">
      <mpv:LastURL>album.pvm</mpv:LastURL>
      <mpv:Related mpv:relationship="derivedFrom">
        <mpv:DocumentRef mpv:idRef="DOC001"/>
      </mpv:Related>
      ...
    </mpv:Document>
    ...
  </mpv:AssetList>
</file:Manifest>
```

Related: [MANIFEST]

3.1.17 Locale of the Manifest

PLW100-17 Locale of the Manifest. The <mpv:Document> asset representing the manifest itself SHOULD have the <dc:language> element set to the default locale of the manifest's content.

This practice conforms to [MANIFEST]. The MPV manifest is authored for a single locale, where a locale is a combination of language and territory, such as English-U.S. The locale of a manifest is recorded as a <dc:language> element of the manifest s own <mpv:Document> asset. The default locale of all text in the manifest is identified by the <dc:language> element within the <mpv:Document> element representing the current manifest. A player MAY honor the use of <dc:language> for purposes of sorting, line wrapping, currency, and other values.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest
       xmlns:file="http://ns.osta.org/manifest/1.0/"
       xmlns:file2="http://ns.osta.org/manifest/2.0"
       xmlns:mpv="http://ns.osta.org/mpv/1.0/"
       xmlns:mpvp="http://ns.osta.org/mpv/presentation/1.0/"
       xmlns:mpvm="http://ns.osta.org/mpv/music/1.0/"
       xmlns:nmf="http://ns.osta.org/nmf/1.0/">
  ...
  <mpv:AssetList>
    <mpv:Document mpv:id="DOC001">
      <mpv:LastURL>metahist/hist0001.pvm</mpv:LastURL>
      <nmf:Metadata>
        <dc:Properties>
          <dc:language>en-US</dc:language>
          ...
        </dc:Properties>
      </nmf:Metadata>
      ...
    </mpv:Document>
    ...
  </mpv:AssetList>
</file:Manifest>
```

For the locale of each asset, see Section 3.1.8.

Related: [MANIFEST]

3.1.18 <mpv:InstanceID> element

PLW100-18 <mpv:InstanceID> element. A Writer SHOULD use the <mpv:InstanceID> element to represent the manifest itself if possible.

This practice is a refinement of [MANIFEST]. <mpv:InstanceID> is a unique number for every manifest. A sophisticated Writer SHOULD use it for easy identification of the manifest. A poor device that cannot generate a unique number MAY ignore this element.

Example:

```
<mpv:AssetList>
  <mpv:Document mpv:id="DOC001">
    <mpv:InstanceID>urn.osta-
org.mpv.uuid.234562234BDF9BBA934338DFBFFDE8342</mpv:InstanceID>
    <mpv:LastURL>metahist/hist0001.pvm</mpv:LastURL>
    <nmf:Metadata>
      <dc:Properties>
        <dc:language>en-US</dc:language>
        ...
      </dc:Properties>
    </nmf:Metadata>
    ...
  </mpv:Document>
  ...
</mpv:AssetList>
```

Related: [MANIFEST]

3.1.19 "mpv:filesystem" attribute

PLW100-19 <mpv:LastURL> "mpv:filesystem" attribute. Each asset that has at least one <mpv:LastURL> element SHOULD provide at least one <mpv:LastURL> element for each file system of the storage media on which the MPV playlist is resident and with which the asset can be accessed.

This practice is to guarantee that a Reader can find a <mpv:LastURL> element with the "mpv:filesystem" attribute that corresponds to the active file system. However, this requirements does not mean that all <mpv:LastURL> elements must have a "mpv:filesystem" attribute.

The "mpv:filesystem" attribute is used on <mpv:LastURL> elements to identify the file system with which the LastURL value is associated. The LastURL value can be different for different file systems because different file systems have limits such as the number of characters in each pathname segment. For CDs in particular, it is commonplace to have multiple file systems on a CD, such as the ISO-9660-1 file system and the Joliet file system.

See also Section 3.1.21 and 3.1.22.

3.1.20 UTF-8 and URL-encoding

PLW100-20 <mpv:LastURL> UTF-8 and URL-encoded values. The value of a <mpv:LastURL> element MUST be UTF-8-encoded and URL-encoded.

This practice is to guarantee that a LastURL value can be properly decoded.

A LastURL value is always UTF-8 and URL-encoded. The algorithm for generating a LastURL value is as follows:

1. UTF-8 encode the pathname.
   a. Converts multi-byte or Unicode strings into UTF-8 strings
   b. UTF-8 encoding is required by MPV Manifest
   c. Note that file system-specific practices are deprecated, e.g. MS-DOS and MS-Windows directory separator backslashes "\" and Macintosh directory separators colons ":". This is not appropriate even if the target file system of the LastURL value is the same as the file system from which the pathname was derived. Hence, when converting between file systems, e.g. from a hard disk file system like NTFS to a CD file system like ISO9660-1, additional processing may be needed to convert directory separators and pathname components to be compliant with the target file system.
2. URL-encode the pathname.
   a. Converts all URL-protected characters into equivalents, such as space " " into %20.

The resulting value should be transportable and processable in any environment, since is uses only a 7-bit character set.

To convert the LastURL value back to a value that can be used by the file system, reverse the algorithm.

1. URL-decode the pathname.
2. UTF-8 decode the pathname.

The resulting value should be correct for the given file system indicated by the "mpv:filesystem" attribute.

See also Section 3.1.21, 3.1.22.

3.1.21 Relative Form of URI

PLW100-21 A Writer SHOULD always make use of a relative form of URI.

This practice guarantees that applications can import, embed or move an MPV album to another storage unit without heavy processing. Using a relative form of URI guarantees the least amount of processing during those operations.

There are two forms of URI: an absolute URI and a relative URI. A Writer SHOULD avoid using an absolute URI if a relative URI is available. See [URI] for more details.

The *current context* of the URI is the current context (current directory) which the MPV manifest reside.

Example:

```
<mpv:Still>
  <mpv:LastURL mpv:filesystem="ISO9660-1">../IMG/BAR001.JPG</mpv:LastURL>
  ...
</mpv:Still>
```

Example:

```
<mpv:Still>
  <mpv:LastURL mpv:filesystem="Unix">./images/home.jpg</mpv:LastURL>
  ...
</mpv:Still>
```

Reference: Section 1.4 and Section 3 of [URI].

3.1.22 URI Component Delimiter

PLW100-22 A Writer MUST use the "/" (slash) character as component (directory) delimiter to specify a <mpv:LastURL>.

This practice is a clarification of the syntax of <mpv:LastURL>. [URI] defines that a component (directory) delimiter is denoted by a "/" (slash) character.

Note that the "filesystem" attribute of the <mpv:LastURL> element cannot override this requirement. For example, all Windows file systems (e.g. "FAT16", "FAT32", "Windows" file system attributes) use the "\" (backslash) character as a directory separator. If the location of <mpv:Still> resides in a FAT32 file system that is ".\images\still001.jpg", the <mpv:LastURL> MUST be like this:

```
<mpv:LastURL filesystem="FAT32">./images/still001.jpg</mpv:LastURL>
```

Reference: Section 1.4 of [URI], Section 5.7 of [MPVCore].

3.1.23 Relationship between metadata formats.

Although MPV is very flexible to embed any kind of custom extensions, there are few restrictions. Currently MPV makes use of following well-known metadata formats:

- Dublin Core -- Normalized Metadata Format [DC-NMF]
- JPEG2000 Metadata Format, [J2K-Part2]
- MPV Music Profile Metadata, [MPVMusic]

PLW100-23 Custom Metadata format always have lower priority than predefined metadata formats such as [DC-NMF], [J2K-Part2], or [MPVMusic]. An application MUST use well-known metadata formats such as [DC-NMF], [J2K-Part2] or [MPVMusic] in preference to their own metadata format. They MAY use their own metadata format if and only if there is no way to specify such information in a required metadata format.

This practice guarantees that applications read common information without supporting particular metadata format or syntax. Using custom metadata is encouraged if and only if certain metadata cannot be represented by using well-known metadata format such as [DC-NMF], [J2K-Part2], or [MPVMusic].

For example, if the application wants to specify the title information of a media file, the application must provide a <dc:title> element prior to using their own metadata format since [DC-NMF] already defines how to write plain-text title information.

Example:

```
<mpv:Still mpv:id="STILL001">
  ...
  <nmf:Metadata>
    <dc:Properties>
      <dc:title>Happiest Memory</dc:title>
      ...
    </dc:Properties>
  </nmf:Metadata>
  <mpv:Metadata>
    <Properties xmlns="http://some.vendors.com/ns/1.0/">
      <title><emph><strong>H</strong>appiest <strong>M</strong>emory</emph></title>
    </Properties>
  </mpv:Metadata>
</mpv:Still>
```

See also the example of Section 3.1.6.

3.1.24 <mpv:Metadata> for custom metadata

PLW100-24 To embed custom metadata into the MPV manifest, applications SHOULD use <mpv:Metadata> elements for existing schema and <nmf:Metadata> for new schema.

The element, <nmf:Metadata> can be used if the metadata format is conforming to the NMF specification. New metadata schema SHOULD be defined following the rules of the NMF specification, as this format realizes additional benefits to applications. However, for existing (non NMF) metadata schema, <mpv:Metadata> MUST be used.

Example of embedded DIG35 metadata:

```
<mpv:Metadata mpv:schemaURI="http://www.digitalimaging.org/dig35/1.1/xml">
  <METADATA xmlns="http://www.digitalimaging.org/dig35/1.1/xml">
    <BASIC_IMAGE_PARAM>
      <BASIC_IMAGE_INFO>
        <IMAGE_SIZE>
          <WIDTH>1600</WIDTH>
          <HEIGHT>1200</HEIGHT>
        </IMAGE_SIZE>
      </BASIC_IMAGE_INFO>
    </BASIC_IMAGE_PARAM>
  </METADATA>
</mpv:Metadata>
```

Example of embedded native XMP metadata:

```
<mpv:Metadata mpv:schemaURI="adobe:ns:meta/">
  <x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="XMPTk 2.8">
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
      <rdf:Description about="" xmlns:xmp="http://ns.adobe.com/xap/1.0/"
          xmp:Author="Pieter van Zee"
          xmp:CreateDate="2002-03-25T21:07:00Z">
      </rdf:Description>
    </rdf:RDF>
  </x:xmpmeta>
</mpv:Metadata>
```

Reference: Section 7.1 and Section 7.2 of [MPVCore].

3.1.25 Thumbnail rendition

PLW100-25 Thumbnail rendition. An asset MAY have a thumbnail rendition. Thumbnail renditions MUST be simple visual assets, e.g. <mpv:Still> or <mpv:Video>, and the resolution MUST be less than VGA resolution.

Thumbnail renditions are used to allow a Reader to enhance performance and reduce memory usage by utilizing a lower-resolution image than the master asset. For non-visual assets, thumbnail renditions provide visual displays to accompany or stand-in for the non-visual assets. For a mpv:ManifestLink asset, the thumbnail rendition shows a proxy image or video asset for the referenced manifest.

Example:

```
<mpv:Still mpv:id="STILL001">
  ...
  <mpv:Rendition mpv:renditionUsage="thumbnail">
    <mpv:StillRef mpv:idRef="THUMB001"/>
  </mpv:Rendition>
</mpv:Still>
```

3.1.26 Screen rendition

PLW100-26 Screen rendition. Any asset MAY have a screen rendition. Screen renditions MUST be visual assets, e.g. <mpv:Still>, <mpv:StillMultishotSequence>, <mpv:StillPanoramaSequence>, <mpv:Video>, and the resolution MUST be at least corresponding to typical screen resolution among current products, such as VGA or 1MPixel 2D resolution.

Screen renditions are used for multiple purposes. They allow a Reader to enhance performance and reduce memory usage by utilizing a lower-resolution still image than the master asset. For non-visual assets, screen renditions provide visual displays to accompany or stand-in for the non-visual assets. For a <mpv:ManifestLink> asset, the screen rendition shows a proxy image or video asset for the referenced manifest.

```
<mpv:Still mpv:id="STILL001">
  ...
  <mpv:Rendition mpv:renditionUsage="screen001">
    <mpv:StillRef mpv:idRef="SCREEN001"/>
  </mpv:Rendition>
</mpv:Still>
```

3.1.27 Show rendition

PLW100-27 Show rendition. Every composite asset MAY have a Show rendition and non-visual assets MAY have a show rendition.

Show renditions are used to allow a Reader to enhance its feature set without introducing complicated software enhancements. By using show renditions, a Reader can easily present content formatted for presentation without having to implement complicated presentation logic.

For example, on a device with slow storage and a small amount of memory, it is practically impossible to play the <mpv:StillWithAudio> or <mpvm:AudioWithStills> asset type, since to play these assets, the Reader would have to retrieve at least two media files from the storage media. To provide a slide show playing mode to the user on such a device, the Writer can prepare a video asset which shows like a true <mpv:StillWithAudio> or <mpvm:AudioWithStills>, and the Reader can make use of this video-typed show rendition.

3.1.28 Subsampled rendition

PLW100-28 Subsampled rendition, An asset MAY have a subsampled rendition. A sub-sampled visual rendition asset MUST have different (lower) resolution than the original asset. A sub-sampled audio asset MUST have different (lower) sampling rate than the original asset.

This practice gives the Reader additional choices when seeking to display an asset file.

Example:

```
<mpv:Still mpv:id="ID000001">
  ...
  <mpv:Rendition mpv:renditionUsage="subsampled">
    <mpv:StillRef mpv:idRef="ID000001S"/>
  </mpv:Rendition>
  <mpv:Rendition mpv:renditionUsage="thumbnail">
    <mpv:StillRef mpv:idRef="ID000001T"/>
  </mpv:Rendition>
</mpv:Still>
<mpv:Still mpv:id="ID000001S">
  ...
  <mpv:LastURL mpv:filesystem="ISO9660-1">SCREEN/IMG0003.JPG</mpv:LastURL>
  ...
```

```
</mpv:Still>
<mpv:Still mpv:id="ID000001T">
  ...
  <mpv:LastURL mpv:filesystem="ISO9660-1">THUMB/IMG0003.JPG</mpv:LastURL>
  ...
</mpv:Still>
```

If the top-level asset format cannot be understood and if it has two renditions: an "alt" rendition and a "sub-sampled" rendition, the priority of renditions is implementation defined.

3.1.29 Attribute "mpv:manifestLinkIdRef" NOT USED

PLW100-31 Attribute "mpv:manifestLinkIdRef". The attribute "mpv:manifestLinkIdRef" MUST NOT be used on any element.

By disallowing use of "mpvp:manifestLinkIdRef" attribute, all the assets that a given MPV manifest references are contained in that manifest. This allows applications to work very efficiently since they don't need to search another MPV manifest. For example, if the one MPV manifest heavily uses references to another MPV manifest using "mpv:manifestLinkIdRef" attribute, it is very difficult to import, export, move, delete, or insert any part of these MPV manifests to another storage device. It will very likely produce many broken references. In addition, not all applications are powerful enough to read multiple MPV manifests simultaneously.

In addition, all the assets referenced by the <mpvp:Album> must be present in the <mpv:AssetList> within the same MPV manifest. This is enforced by disallowing use of the "mpvp:manifestLinkIdRef" attribute on any asset reference element.

As a result, any asset referencing MUST be done only within one MPV manifest.

[Forward Reference] <mpvp:Album> (Chapter 4)

3.2 Reader Requirements & Guidelines

Applications which read and give the MPV experience to the user should follow all the requirements and guidelines in this section.

3.2.1 File system based Asset Access

PLR100-1 File system based Asset Access. The appropriate storage media file system MUST be used to access asset files referenced by MPV assets. Direct addressing to fixed locations on the storage media is not supported.

Since the only way to specifying the location of each asset is to use <mpv:LastURL> element, MPV does not use direct addressing to fixed locations on the storage media. In other words, to support MPV technology, the target system MUST have file systems, or at least support file system based access.

Example:

```
<mpv:Still mpv:id="ID000101">
  <mpv:LastURL mpv:filesystem="ISO9660-1">DSC01934.JPG</mpv:LastURL>
  ...
</mpv:Still>
```

3.2.2 Respect <file:Profile> information.

PLR100-2 A Reader SHOULD respect the <file:Profile> information.

This practice guarantees that users see a consistent experience. Although it is possible to determine the usage of MPV Profiles by analyzing the whole document, a Reader SHOULD give users an experience which is provided in <file:Profile> elements form.

In other words, if the document contains <mpvp:Album> elements and provides no MPV Presentation Profile <file:Profile> element, and provides only MPV Basic Profile <file:Profile> element, Readers SHOULD consider the document's Profile usage as the MPV Basic Profile.

For example, consider the following MPV document fragment:

```
<file:Manifest ...>
    xmlns:file="http://ns.osta.org/manifest/1.0/"
    xmlns:mpv="http://ns.osta.org/mpv/1.0/"
    xmlns:mpvp="http://ns.osta.org/mpv/presentation/1.0/"
    xmlns:mpvm="http://ns.osta.org/mpv/music/1.0/"
    xmlns:nmf="http://ns.osta.org/nmf/1.0/"
    ...>
  <nmf:Metadata>
    <file:ManifestProperties>
      <ProfileBag xmlns="http://ns.osta.org/manifest/1.0/">
        <Profile>http://ns.osta.org/mpv/basic/is/1.0/</Profile>
      </ProfileBag>
    </file:ManifestProperties>
    ...
  </nmf:Metadata>
  <mpvp:Album>
    <mpvp:Foreground>
      ...
    </mpvp:Foreground>
  </mpvp:Album>
  <mpv:AssetList>
    <mpv:Still>
      ...
    </mpv:Still>
    <mpv:Audio>
      <nmf:Metadata>
        <mpvm:MusicProperties>
          <mpvm:PrincipleArtist>...</mpvm:PrincipleArtist>
          ...
        </mpvm:MusicProperties>
        ...
      </nmf:Metadata>
      ...
    </mpv:Audio>
    ...
  </mpv:AssetList>
</file:Manifest>
```

Although this manifest uses the MPV Presentation Profile (by providing the <mpvp:Album> element), and the MPV Music Profile (by providing the <mpvm:MusicProperties> element), Readers SHOULD consider the manifest's usage of Profiles as MPV Basic Profile since only the Basic Profile <file:Profile> element is provided.

3.2.3 Fragment Identifier Processing

PLR100-3 <mpv:LastURL> Fragment Identifier Processing. Fragment identifiers in <mpv:LastURL> values MUST be allowed and handled gracefully, even if the specific fragment identifier couldn't be understood.

This practice guarantees that techniques such as identifying thumbnails resident in JPEG Exif 2.1+ files using fragment identifiers will be supported by MPV players. If a specific fragment identifier is not understood, the player may choose to process the asset as an unknown type.[4]

Example:

```
<mpv:Still mpv:id="ID000100">
  <mpv:LastURL>IMG001.JPG</mpv:LastURL>
  <mpv:Rendition mpv:renditionUsage="thumbnail">
    <mpv:StillRef mpv:idRef="ID000200"/>
  </mpv:Rendition>
</mpv:Still>
...
<mpv:Still mpv:id="ID000200">
  <mpv:InstanceID>AC937BCFA3B340da971BAF09B17DBC324</mpv:InstanceID>
  <mpv:LastURL
      filesystem="ISO9660-1">IMG001.JPG#vnd.osta-org.exif-thumb</mpv:LastURL>
  <nmf:Metadata>
    <dc:Properties>
      <dc:format>image/vnd.osta-org.exif-thumb</dc:format>
      ...
    </dc:Properties>
  </nmf:Metadata>
  ...
</mpv:Still>
```

3.2.4 Thumbnail rendition

PLR100-4 Thumbnail rendition. A Reader SHOULD display thumbnail renditions of assets when the Reader device or application can do visual display at thumbnail resolutions. When they are present, thumbnail renditions MUST be displayed for the following asset types: <mpv:Still>, <mpv:Video>, <mpv:ManifestLink>.

Example:

```
<mpv:Still mpv:id="ID000001">
...
  <mpv:Rendition mpv:renditionUsage="screen">
    <mpv:StillRef mpv:idRef="ID000001S"/>
  </mpv:Rendition>
  <mpv:Rendition mpv:renditionUsage="thumbnail">
    <mpv:StillRef mpv:idRef="ID000001T"/>
  </mpv:Rendition>
</mpv:Still>
<mpv:Still mpv:id="ID000001S">
...
  <mpv:LastURL mpv:filesystem="ISO9660-1">SCREEN/IMG0003.JPG</mpv:LastURL>
...
</mpv:Still>
<mpv:Still mpv:id="ID000001T">
...
  <mpv:LastURL mpv:filesystem="ISO9660-1">THUMB/IMG0003.JPG</mpv:LastURL>
...
</mpv:Still>
```

4 Note that this requirement forces the application to handle fragment identifiers correctly, not to understand them.

3.2.5 Screen rendition

PLR100-5 Screen rendition. A Reader SHOULD display screen renditions of assets when the Reader device or application can do visual display at screen resolutions. When they are present, screen renditions MUST be displayed for the following asset types: <mpv:Still>, <mpv:Video>, <mpv:Audio>, <mpv:ManifestLink>.

See also 3.1.25 for the example.

3.2.6 Show rendition

PLR100-6 Show rendition. A Reader MAY display the Show rendition of assets when the Reader device or application can do visual display at presentation rendition.

3.2.7 Alt rendition

PLR100-7 Renditions: alt. Readers SHOULD first try to use the primary asset referenced by the <mpv:LastURL> value. However, when the asset cannot be found or the asset file format cannot be understood, the Reader SHOULD use one of the alt Renditions, if present.

This practice gives the Reader additional choices when seeking to display an asset file.

Example:

```
<mpv:Still mpv:id="ID000001">
  ...
  <mpv:Rendition mpv:renditionUsage="screen">
    <mpv:StillRef mpv:idRef="ID000001S"/>
  </mpv:Rendition>
  <mpv:Rendition mpv:renditionUsage="thumbnail">
    <mpv:StillRef mpv:idRef="ID000001T"/>
  </mpv:Rendition>
  <mpv:Rendition mpv:renditionUsage="alt">
    <mpv:StillRef mpv:idref=”ID000001A"/>
  </mpv:Rendition>
</mpv:Still>
<mpv:Still mpv:id="ID000001S">
  <mpv:LastURL mpv:filesystem="ISO9660-1">SCREEN/IMG0003.JPG</mpv:LastURL>
  ...
</mpv:Still>
<mpv:Still mpv:id="ID000001T">
  <mpv:LastURL mpv:filesystem="ISO9660-1">THUMB/IMG0003.JPG</mpv:LastURL>
  ...
</mpv:Still>
<mpv:Still mpv:id="ID000001A">
  ...
  <mpv:LastURL mpv:filesystem='ISO9660-1">ALT/IMG0003.PNG</mpv:LastURL>
</mpv:Still>
```

If the top-level asset format cannot be understood, and if it has two renditions: an "alt" rendition and a "sub-sampled" rendition, the priority of renditions is implementation defined.

See also Section 3.1.28.

3.2.8 Sub-sampled rendition

PLR100-8 Sub-sampled rendition. Readers SHOULD first try to use the primary asset referenced by the <mpv:LastURL> value. However, when the asset cannot be found or the asset file format cannot

be understood, the Reader SHOULD use one of the sub-sampled renditions, if present.

See also Section 3.2.7.

3.2.9 Render all top-level assets

PLR100-9 A Reader SHOULD render all top-level assets to the user. The rendering order is the sequence in which the top-level assets occur in the <mpv:AssetList>.

This practice guarantees that users can see all the information placed in the MPV document. Section 6.1 of [MPVCore] states that <mpv:AssetList> is unordered, and only [MPVPres] provides a format mechanism to impose presentation content and order.

However, in practice, it is very useful to be able to rely on a presentation sequence for manifests compliant with the MPV Basic Profile. The MPV Interoperability Specification determines that the presentation sequence of a <mpv:AssetList> in the absence of any other presentation information, and SHOULD be rendered to preserve the first-to-last ordering of top-level assets within the <mpv:AssetList>.

Reference: Section 6.1 of [MPVCore].

3.3 Best Practices

3.3.1 Writer Practices

Writer applications should write valid, efficient MPV manifest.

When importing or exporting existing MPV manifests to or from the removable media (e.g. CD, DVD, a memory card), the Writer application should take great care of transforming the <mpv:LastURL> elements of all assets. In other words, when moving an asset to a media with a different file system, the Writer application may need to change the "mpv:filesystem" attribute of the <mpv:LastURL> element for each asset, and may need to convert the URI properly.

If a different Writer writes a MPV manifest, it is strongly recommended that the application leave the custom metadata in the MPV manifest even though the application cannot understand that. This helps the original application can work on modified version of MPV manifest later.

If the storage device has sufficient space, it is recommended to generate various rendition assets: thumbnail, screen, alt, or sub-sampled renditions to help Reader applications.

It is recommended not to write an empty element that has no useful information. For example, the most important child elements of the <mpvp:Album> are the <mpvp:Foreground> and <mpvp:Background> elements. If the manifest has no background asset information, it is better to write the <mpvp:Foreground> element only. For example:

```
<mpvp:Album>
  <mpvp:Foreground>
    <mpv:StillRef mpv:id="still001"/>
    ...
  </mpvp:Foreground>
</mpvp:Album>
```

A bad example might be:

```
<mpvp:Album>
  <mpvp:Foreground>
    <mpv:StillRef mpv:id="still001"/>
```

```
    ...
  </mpvp:Foreground>
  <mpvp:Background>
  </mpvp:Background>
</mpvp:Album>
```

When specifing URI of profiles, or the URI of namespaces, applications are encouraged to use the URI values contained in this document. When writing those URI values, do not omit the component delimiter, "/" (slash character) unless told otherwise.

Example:

```
<file:Manifest
       xmlns:file="http://ns.osta.org/manifest/1.0/"
       xmlns:file2="http://ns.osta.org/manifest/2.0"
       xmlns:mpv="http://ns.osta.org/mpv/1.0/"
       xmlns:nmf="http://ns.osta.org/nmf/1.0/"
       xmlns:dc="http://purl.org/dc/elements/1.1/"
       xmlns:dcterms=http://purl.org/dc/terms/"
       xmlns:jpx="http://www.jpeg.org/jpx/">
  ...
</file:Manifest>
```

Bad example:

```
<file:Manifest
       xmlns:file="http://ns.osta.org/manifest/1.0/"
       xmlns:file2="http://ns.osta.org/manifest/2.0"
       xmlns:mpv="http://ns.osta.org/mpv/1.0/"
       xmlns:nmf="http://ns.osta.org/nmf/1.0"
       xmlns:dc="http://purl.org/dc/elements/1.1/"
       xmlns:dcterms=http://purl.org/dc/terms"
       xmlns:jpx="http://www.jpeg.org/jpx/">
  ...
</file:Manifest>
```

Applications that modify existing MPV documents are encouraged to follow the practice described in [MANIFEST].

It is possible that a manifest declares only the Basic Profile IS Profile identifiers in it. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest ...>
  <nmf:Metadata>
    <file:ManifestProperties>
      <file:ProfileBag>
        <file:Profile>http://ns.osta.org/mpv/basic/is/1.0/</file:Profile>
      </file:ProfileBag>
    </file:ManifestProperties>
    ...
  </nmf:Metadata>
</file:Manifest>
```

However, the Writer SHOULD declare MPV Basic Profile identifier together. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest ...>
```

```
  <nmf:Metadata>
    <file:ManifestProperties>
      <file:ProfileBag>
        <file:Profile>http://ns.osta.org/mpv/basic/1.0/</file:Profile>
        <file:Profile>http://ns.osta.org/mpv/basic/is/1.0/</file:Profile>
      </file:ProfileBag>
    </file:ManifestProperties>
    ...
  </nmf:Metadata>
</file:Manifest>
```

The reason for this recommendation is to keep the compatibility between IS compliant applications and legacy applications (which do not recognize IS Profile identifiers).

If the manifest uses features that are non-compliant with the MPV Basic Profile, the manifest MUST NOT contain the IS compliant identifier. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest ...>
  <nmf:Metadata>
    <file:ManifestProperties>
      <file:ProfileBag>
        <file:Profile>http://ns.osta.org/mpv/basic/1.0/</file:Profile>
      </file:ProfileBag>
    </file:ManifestProperties>
    ...
  </nmf:Metadata>
</file:Manifest>
```

3.3.2 Reader Practices

The basic browsing experience should provide two basic capabilities:

- Browsing of thumbnails/list view of photo-video-music contents.
- Browsing of full-screen views of photo-video-music contents.

To get better performance, an application may use thumbnail or screen rendition depending on the context. An advanced browsing application MAY treat composite assets such as <mpv:StillWithAudio>, <mpv:StillMultishotSequence>, <mpv:StillPanoramaSequence>, <mpv:Seq>, <mpv:Par>, and <mpvm:AudioWithStills> specially. For example, the application could show an iconic representation indicating that the thumbnail represents a certain kind of composite asset.

Since MPV doesn't require specific media file formats, readers might confront a MPV manifest that has unsupported asset/media types. In this case, applications can take following actions:

- Ignore the unsupported asset type gracefully.
- Show blank image/video/audio to acknowledge users that this media type is unsupported. (Highly recommended)

In the former case, it is recommended that the application acknowledge this to the user in some ways so that the user knows that there's unsupported media present.

When a Reader tries to detect the usage of MPV T profile, it SHOULD try to match the manifest one of the following combination of Profile declarations:

- Case A: The manifest contains only the T Profile identifier.
- Case B: The manifest contains only IS-compliant T Profile identifiers.

- Case C: The manifest contains both Profile identifiers and IS-compliant Profile identifiers

If the MPV Reader detects case A, the Reader SHOULD try to process the manifest as if the manifest is an IS-compliant manifest. If the manifest does not comply with IS version, the Reader SHOULD try to process it using full feature set of the T profile.

If the MPV Reader detects case B, the Reader SHOULD try to process the manifest with IS-compliant features only. The Reader SHOULD not try to use features beyond IS-compliant T Profile requirements.

If the MPV Reader detects case C, the Reader SHOULD treat the manifest as in case B.

4 Presentation Profile Interoperability Specification

To provide enhanced experience to users, devices can make use of the MPV Presentation Profile. However, if devices are not powerful enough, vendors can ignore the MPV Presentation Profile support on their devices. See also Section 3.1.5 for the possible combinations of current MPV Profiles.

The original MPV Presentation Profile is designed to enhance the functionality of MPV Basic Profile. This means that the MPV Presentation Profile is based on the MPV Basic Profile. Likewise, the requirements in this chapter are designed for the application that already satisfies the requirements in Chapter 3. In other words, an application that supports MPV Presentation Profile Interoperability Specification (this chapter) MUST support MPV Basic Profile Interoperability Specification in Chapter 3.

4.1 Writer Requirements & Guidelines

An application that writes a MPV manifest with MPV Presentation Profile support and the Presentation Profile IS Profile Identifier (See section 4.1.2) MUST follow all the requirements and guidelines in this section.

4.1.1 Support MPV Presentation Profile

PLW100-29 Presentation Profile Declaration. MPV Writers that implement the Presentation Profile MUST declare it in the MPV Manifest in order for MPV Readers to consider MPV Presentation Profile content.

The MPV Presentation Profile Specification describes presentation behavior for MPV playlists. A MPV Writer MUST declare use of the MPV Presentation Profile in order for a MPV application to consider any presentation-specific content.

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest
       xmlns:file="http://ns.osta.org/manifest/1.0/"
       xmlns:file2="http://ns.osta.org/manifest/2.0"
       xmlns:mpv="http://ns.osta.org/mpv/1.0/"
       xmlns:mpvp="http://ns.osta.org/mpv/presentation/1.0/"
       xmlns:mpvm="http://ns.osta.org/mpv/music/1.0/"
       xmlns:nmf="http://ns.osta.org/nmf/1.0/">
  <nmf:Metadata>
    <file:ManifestProperties>
      <file:ProfileBag>
        ...
       <file:Profile>http://ns.osta.org/mpv/presentation/1.0/</file:Profile>
        ...
      </file:ProfileBag>
      <file2:AboutManifestMPVDocumentID>DOC001</file2:AboutManifestMPVDocumentID>
      <file2:WrittenBy>http://www.companyA.com/Device/Model1934/</file2:WrittenBy>
    </file:ManifestProperties>
  </nmf:Metadata>
  ...
  <mpv:AssetList>
    ...
    <mpv:Document mpv:id="DOC001">
      ...
    </mpv:Document>
```

```
    ...
  </mpv:AssetList>
</file:Manifest>
```

4.1.2 Presentation Profile IS Profile identifier

PLW100-30 Declare IS-Compliant Profile. A MPV Manifest that complies with the requirements of the MPV Presentation Profile Interoperability Specification (this section) MUST declare that compliance by listing the Presentation Profile IS identifier in a <file:Profile> entry in the <file:ManifestProperties>.

The MPV Presentation Profile Interoperability Specification profile identifier is:

Interoperability Specification For ...	*IS Profile Identifier*
Presentation Profile	http://ns.osta.org/mpv/presentation/is/1.0/

Use of this IS profile identifier allows a MPV Reader to determine that the manifest complies with the Presentation Profile Interoperability Specifications.

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest
       xmlns:file="http://ns.osta.org/manifest/1.0/"
       xmlns:file2="http://ns.osta.org/manifest/2.0"
       xmlns:mpv="http://ns.osta.org/mpv/1.0/"
       xmlns:mpvp="http://ns.osta.org/mpv/presentation/1.0/"
       xmlns:mpvm="http://ns.osta.org/mpv/music/1.0/"
       xmlns:nmf="http://ns.osta.org/nmf/1.0/">
  <nmf:Metadata>
    <file:ManifestProperties>
      <file:ProfileBag>
        <file:Profile>http://ns.osta.org/mpv/basic/1.0/</file:Profile>
        <file:Profile>http://ns.osta.org/mpv/basic/is/1.0/</file:Profile>
        <file:Profile>http://ns.osta.org/mpv/presentation/1.0/</file:Profile>
        <file:Profile>http://ns.osta.org/mpv/presentation/is/1.0/</file:Profile>
        ...
      </file:ProfileBag>
      <file2:AboutManifestMPVDocumentID>DOC001</file2:AboutManifestMPVDocumentID>
      <file2:WrittenBy>http://www.companyA.com/Device/Model1934/</file2:WrittenBy>
    </file:ManifestProperties>
  </nmf:Metadata>
  ...
  <mpv:AssetList>
    ...
    <mpv:Document mpv:id="DOC001">
      ...
    </mpv:Document>
    ...
  </mpv:AssetList>
</file:Manifest>
```

See also Section 3.1.2.

4.1.3 Album Asset Validation

PLW100-39 <mpv:idRef> references MUST be valid. Any reference element which is contained in the <mpvp:Foreground> element or the <mpvp:Background> element of the <mpvp:Album> element MUST refer to a valid asset in the <mpv:AssetList>.

For example, a MPV Writer MUST NOT generate the following MPV manifest:

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest ...>
  <nmf:Metadata>
    ...
  </nmf:Metadata>

  <mpvp:Album>
    <mpvp:Foreground>
      <mpv:StillRef mpv:idRef="still001"/>
      <mpv:VideoRef mpv:idRef="video001"/>
    </mpvp:Foreground>
  </mpvp:Album>

  <mpv:AssetList>
    <mpv:Document mpv:id="DOC001">
      ...
    </mpv:Document>

    <mpv:Still mpv:id="still001">
      ...
    </mpv:Still>
  </mpv:AssetList>
</file:Manifest>
```

In the above example, <mpv:Foreground> contains <mpv:VideoRef> element whose "mpv:idRef" attribute "video001" does not exist in the <mpv:AssetList>. Such broken references MUST NOT happen in a valid MPV manifest.

4.1.4 Only one <mpvp:Album> element

PLW100-32 A MPV Manifest MAY have one or more <mpvp:Album> elements.

The [MPVPres] specification presents that it is possible to have more than one <mpvp:Album> elements in a MPV manifest. However, only the first <mpvp:Album> elements (called the default album) MAY be utilized by MPV readers. Hence, it is recommended for MPV writers to generate one <mpvp:Album> element per manifest. See also Section 4.2.1.

Reference: Section 5.2 of [MPVPres]

4.1.5 MUST NOT: Use <mpvp:AlbumRef> to link to another Album

PLW100-33 <mpvp:AlbumRef> NOT USED. The <mpvp:AlbumRef> MUST NOT be used.

The use of the <mpvp:AlbumRef> asset reference is not allowed. Instead, use <mpv:ManifestLinkRef>. A <mpv:ManifestLink> asset is used to link one manifest to another.

4.2 Reader Requirements & Guidelines

An application that reads a MPV manifest with MPV Presetation Profile support and the Presentation

Profile IS Profile Identifier (See section 4.1.2) MUST follow all the requirements and guidelines in this section.

4.2.1 Utilize the default album.

PLR100-10 Readers MUST utilize at least the default album (i.e. the first <mpvp:Album> element).

If the manifest contains more than one album, the default album MUST be utilized by the readers. There is no further requirements for the readers to utilize all <mpvp:Album> elements in a MPV manifest. See also Section 4.1.4.

Reference: Section 5.2 of [MPVPres]

4.3 Examples

This is a full example of a manifest that supports [MPVBasic], and [MPVPres].

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest
       xmlns:file="http://ns.osta.org/manifest/1.0/"
       xmlns:file2="http://ns.osta.org/manifest/2.0"
       xmlns:mpv="http://ns.osta.org/mpv/1.0/"
       xmlns:mpv="http://ns.osta.org/mpv/presentation/1.0/"
       xmlns:nmf="http://ns.osta.org/nmf/1.0/"
       xmlns:dc="http://purl.org/dc/elements/1.1/"
       xmlns:dcterms=http://purl.org/dc/terms/"
       xmlns:jpx="http://www.jpeg.org/jpx/">
  <nmf:Metadata>
    <file:ManifestProperties>
      <file:ProfileBag>
        <file:Profile>http://ns.osta.org/mpv/basic/1.0/</file:Profile>
        <file:Profile>http://ns.osta.org/mpv/basic/is/1.0/</file:Profile>
        <file:Profile>http://ns.osta.org/mpv/presentation/1.0/</file:Profile>
        <file:Profile>http://ns.osta.org/mpv/presentation/is/1.0/</file:Profile>
      </file:ProfileBag>
      <file2:AboutManifestMPVDocumentID>DOC001</file2:AboutManifestMPVDocumentID>
      <file2:WrittenBy>http://www.companyA.com/Device/Model1934/</file2:WrittenBy>
    </file:ManifestProperties>
  </nmf:Metadata>

  <mpvp:Album>
    <mpvp:Foreground>
      <mpv:StillRef mpv:idRef="still001"/>
      <mpv:VideoRef mpv:idRef="video001"/>
    </mpvp:Foreground>
  </mpvp:Album>

  <mpv:AssetList>
    <mpv:Document mpv:id="DOC001">
      <mpv:LastURL mpv:filesystem="ISO9660-1">ALBUM.PVM</mpv:LastURL>
      <mpv:LastURL mpv:filesystem="Unix">./album.pvm</mpv:LastURL>
      <mpv:LastURL>./album.pvm</mpv:LastURL>
      <nmf:Metadata>
        <dc:Properties>
          <dc:creator>Company A Application 1.3</dc:creator>
          <dc:description>this is some description</dc:description>
          <dc:language>en-US</dc:language>
          <dc:title>some title</dc:title>
        </dc:Properties>
```

```
      <dcterms:Properties>
        <dcterms:created>2004-03-14</dcterms:created>
      </dcterms:Properties>
    </nmf:Metadata>
  </mpv:Document>

  <mpv:Still mpv:id="still001">
    <mpv:LastURL mpv:filesystem="ISO9660-1">IMG001.JPG</mpv:LastURL>
    <mpv:LastURL mpv:filesystem="Unix">./img01.jpg</mpv:LastURL>
    <mpv:LastURL>./img01.jpg</mpv:LastURL>
    <mpv:Metadata>
      <jpx:BASIC_IMAGE_PARAM>
        <jpx:BASIC_IMAGE_INFO>
          <jpx:IMAGE_SIZE>
            <jpx:WIDTH>1600</jpx:WIDTH>
            <HEIGHT>1200</jpx:HEIGHT>
          </jpx:IMAGE_SIZE>
        </jpx:BASIC_IMAGE_INFO>
      </jpx:BASIC_IMAGE_PARAM>
    </mpv:Metadata>
    <nmf:Metadata>
      <dc:Properties>
        <dc:description>Some description</dc:description>
        <dc:format>image/jpeg</dc:format>
        <dc:title>some title</dc:title>
      </dc:Properties>
    </nmf:Metadata>
    <mpv:Rendition mpv:renditionUsage="thumbnail">
      <mpv:StillRef mpv:idRef="still-st001"/>
    </mpv:Rendition>
  </mpv:Still>

  <mpv:Still mpv:id="still-st001">
    <mpv:LastURL mpv:filesystem="ISO9660-1">THUMB/IMG001.JPG</mpv:LastURL>
    <mpv:LastURL mpv:filesystem="Unix">./thumb/img01.jpg</mpv:LastURL>
    <mpv:LastURL>./thumb/img01.jpg</mpv:LastURL>
    <mpv:Metadata>
      <jpx:BASIC_IMAGE_PARAM>
        <jpx:BASIC_IMAGE_INFO>
          <jpx:IMAGE_SIZE>
            <jpx:WIDTH>120</jpx:WIDTH>
            <HEIGHT>90</jpx:HEIGHT>
          </jpx:IMAGE_SIZE>
        </jpx:BASIC_IMAGE_INFO>
      </jpx:BASIC_IMAGE_PARAM>
    </mpv:Metadata>
    <nmf:Metadata>
      <dc:Properties>
        <dc:format>image/jpeg</dc:format>
      </dc:Properties>
    </nmf:Metadata>
  </mpv:Still>

  <mpv:Video mpv:id="video001">
    <mpv:LastURL mpv:filesystem="ISO9660-1">vid001.MPG</mpv:LastURL>
    <mpv:LastURL mpv:filesystem="Unix">./vid01.mpg</mpv:LastURL>
    <mpv:LastURL>./vid01.mpg</mpv:LastURL>
    <nmf:Metadata>
      <dc:Properties>
        <dc:description>Some description</dc:description>
        <dc:format>video/mpeg</dc:format>
```

```
          <dc:title>some title</dc:title>
        </dc:Properties>
      </nmf:Metadata>
      <mpv:Rendition mpv:renditionUsage="thumbnail">
        <mpv:StillRef mpv:idRef="still-vt001"/>
      </mpv:Rendition>
    </mpv:Video>

    <mpv:Still mpv:id="still-vt001">
      <mpv:LastURL mpv:filesystem="ISO9660-1">THUMB/IMG002.JPG</mpv:LastURL>
      <mpv:LastURL mpv:filesystem="Unix">./thumb/img02.jpg</mpv:LastURL>
      <mpv:LastURL>./thumb/img02.jpg</mpv:LastURL>
      <mpv:Metadata>
        <jpx:BASIC_IMAGE_PARAM>
          <jpx:BASIC_IMAGE_INFO>
            <jpx:IMAGE_SIZE>
              <jpx:WIDTH>120</jpx:WIDTH>
              <HEIGHT>90</jpx:HEIGHT>
            </jpx:IMAGE_SIZE>
          </jpx:BASIC_IMAGE_INFO>
        </jpx:BASIC_IMAGE_PARAM>
      </mpv:Metadata>
      <nmf:Metadata>
        <dc:Properties>
          <dc:format>image/jpeg</dc:format>
        </dc:Properties>
      </nmf:Metadata>
    </mpv:Still>
  </mpv:AssetList>
</file:Manifest>
```

5 Music Profile Interoperability Specification

Applications can make use of the MPV Music Profile Specification to enhance the usage of music related metadata and <mpvm:AudioWithStills> assets. As with the MPV Presentation Profile Specification, implementers may ignore or support MPV Music Profile Specification.

Originally, the MPV Music Profile can be supported by the application that does not support MPV Presentation Profile. Likewise, an application that supports MPV Music Profile Interoperability Specification (this chapter) MUST support MPV Basic Profile Interoperability Specification in Chapter 3. If that application wants to support MPV Presentation Profile, it MUST also support MPV Presentation Profile Interoperability Specification in Chapter 4.

5.1 Writer Requirements & Guidelines

An application that writes a MPV manifest with MPV Music Profile support and the Music Profile IS Profile identifier (See section 5.1.2) MUST follow all of the requirements and guidelines in this section.

5.1.1 Support MPV Music Profile

PLW100-34 Music Profile Declaration. MPV Writers that implement the MPV Music Profile MUST declare it in the MPV Manifest in order for MPV Readers to consider MPV Music Profile content.

The MPV Music Profile Specification describes music-centric presentation behavior for MPV playlists. A MPV Writer must declare use of the MPV Music Profile in order for a MPV application to consider any music-centric presentation content.

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest
       xmlns:file="http://ns.osta.org/manifest/1.0/"
       xmlns:file2="http://ns.osta.org/manifest/2.0"
       xmlns:mpv="http://ns.osta.org/mpv/1.0/"
       xmlns:mpvp="http://ns.osta.org/mpv/presentation/1.0/"
       xmlns:mpvm="http://ns.osta.org/mpv/music/1.0/"
       xmlns:nmf="http://ns.osta.org/nmf/1.0/"
       ...
  >
  <nmf:Metadata>
    <file:ManifestProperties>
      <file:ProfileBag>
        <file:Profile>http://ns.osta.org/mpv/basic/1.0/</file:Profile>
        <file:Profile>http://ns.osta.org/mpv/music/1.0/</file:Profile>
        ...
      </file:ProfileBag>
      <file2:AboutManifestMPVDocumentID>DOC001</file2:AboutManifestMPVDocumentID>
      <file2:WrittenBy>http://www.companyA.com/Device/Model1934/</file2:WrittenBy>
    </file:ManifestProperties>
  </nmf:Metadata>
  ...
  <mpv:AssetList>
    ...
    <mpv:Document mpv:id="DOC001">
      ...
```

```
    </mpv:Document>
    ...
  </mpv:AssetList>
</file:Manifest>
```

5.1.2 Music Profile IS Profile Identifier

PLW100-35 Declare IS-compliant Profile. A MPV Manifest that complies with the requirements of the MPV Music Profile Interoperability Specification (this section) MUST declare that compliance by listing the Music Profile IS Profile identifier in a <file:Profile> entry in the <file:ManifestProperties>.

The MPV Music Profile Interoperability Specification profile identifier is:

Interoperability Specification For ...	*IS Profile Identifier*
Music Profile	http://ns.osta.org/mpv/music/is/1.0/

Use of this IS profile identifier allows a MPV Reader to determine that the manifest complies with the Music Profile Interoperability Specifications.

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest
       xmlns:file="http://ns.osta.org/manifest/1.0/"
       xmlns:file2="http://ns.osta.org/manifest/2.0"
       xmlns:mpv="http://ns.osta.org/mpv/1.0/"
       xmlns:mpvp="http://ns.osta.org/mpv/presentation/1.0/"
       xmlns:mpvm="http://ns.osta.org/mpv/music/1.0/"
       xmlns:nmf="http://ns.osta.org/nmf/1.0/">
  <nmf:Metadata>
    <file:ManifestProperties>
      <file:ProfileBag>
        <file:Profile>http://ns.osta.org/mpv/basic/1.0/</file:Profile>
        <file:Profile>http://ns.osta.org/mpv/basic/is/1.0/</file:Profile>
        <file:Profile>http://ns.osta.org/mpv/music/1.0/</file:Profile>
        <file:Profile>http://ns.osta.org/mpv/music/is/1.0/</file:Profile>
        ...
      </file:ProfileBag>
      <file2:AboutManifestMPVDocumentID>DOC001</file2:AboutManifestMPVDocumentID>
      <file2:WrittenBy>http://www.companyA.com/Device/Model1934/</file2:WrittenBy>
    </file:ManifestProperties>
  </nmf:Metadata>
  ...
  <mpv:AssetList>
    ...
    <mpv:Document mpv:id="DOC001">
      ...
    </mpv:Document>
    ...
  </mpv:AssetList>
</file:Manifest>
```

See also Section 3.1.2, 4.1.2.

5.1.3 Music Plain-text elements

PLW100-36 mpvm: PrincipleArtist, Genre, AlbumTitle. When information corresponding to any or all

of this basic set of defined MPV Music Profile Specification elements is known, the appropriate elements SHOULD be used to represent the information. If present, an element value MUST be UTF8-encoded plain text and has no specific interpretation other than as human-readable plain text that may be displayed or searched.

This practice is to guarantee that a Reader can read music information regardless of its ability to read the information from the media file. Also, by caching the value in the MPV manifest, it protects that value from loss if the asset itself should be edited. The element value is arbitrary with no specified interpretation other than as a human-readable plain-text string suitable for display or search.

Many applications may also have formatted versions of this content. MPV does not define the interchange of formatted content. The recommended approach is to store the formatted content using metadata extensions while recording the interchangeable form of the information as DC metadata or MPV Music Profile metadata.

Example:

```
<mpv:Audio mpv:id="02-GREAT-SWING-CLASSICS.MP3-20021202031833-a">
  <mpv:LastURL mpv:filesystem="Joliet">Harlem%20Air%20Shaft.mp3</mpv:LastURL>
  <mpv:LastURL mpv:filesystem="ISO9660-1">DUKE_ELL.MP3</mpv:LastURL>
  ...
  <nmf:Metadata>
    <dc:Properties>
      <dc:creator>Duke Ellington and his Orchestra</dc:creator>
      <dc:format>audio/mpeg</dc:format>
      <dc:title>Harlem Air Shaft</dc:title>
      ...
    </dc:Properties>
    <mpvm:MusicProperties>
      <mpvm:AlbumTitle>Great SWING CLASSICS in HI-FI</mpvm:AlbumTitle>
      <mpvm:Genre>Jazz</mpvm:Genre>
      <mpvm:MusicBy>Duke Ellington</mpvm:MusicBy>
      <mpvm:NumTracks>14</mpvm:NumTracks>
      <mpvm:PlayingTime>234.12</mpvm:PlayingTime>
      <mpvm:PrincipalArtist>Duke Ellington</mpvm:PrincipalArtist>
      <mpvm:Recorded>1955-11-17</mpvm:Recorded>
      ...
    </mpvm:MusicProperties>
  </nmf:Metadata>
  ...
</mpv:Audio>
```

See also Section 3.1.6.

5.1.4 <mpvm:PlayingTime> element

PLW100-37 <mpvm:PlayingTime> SHOULD be used when the playing time is known. If present, the value MUST be encoded as specified in [MPVMusic].

This practice guarantees that the Reader can determine the playing time of a given audio asset without opening the media file. This is important on low-performance systems.

Example:

```
<mpv:Audio mpv:id="ID000002">
  <nmf:Metadata>
    <dc:Properties>
      <dc:format>audio/mpeg</dc:format>
      <dc:title>Beach Boys</dc:title>
      ...
```

```
    </dc:Properties>
    <mpvm:MusicProperties>
      <mpvm:AlbumTitle>Made In USA</mpvm:AlbumTitle>
      <mpvm:PlayingTime>119.96</mpvm:PlayingTime>
      ...
    </mpvm:MusicProperties>
  </nmf:Metadata>
  <mpv:LastURL mpv:filesystem="ISO9660-1">409D.MP3</mpv:LastURL>
  ...
</mpv:Audio>
```

5.1.5 Genre Mapping

PLW100-38 Applications that support MPV Music Profile SHOULD use the predefined genre identifiers that are defined in [MPVMusic].

This practice guarantees the mapping between various metadata formats such as ID3v1, ID3v2 is straightforward to implement, and guarantees that users get a consistent genre name between applications.

Reference: Section 7.1.1 of [MPVMusic].

5.2 Reader Requirements & Guidelines

Currently, there are no specific requirements for MPV Music Profile-aware Readers. Reader requirements may be added in a future revision of this document.

5.3 Example

This is a full example of a manifest that supports MPV IS version of [MPVBasic] and [MPVMusic].

```
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest
       xmlns:file="http://ns.osta.org/manifest/1.0/"
       xmlns:file2="http://ns.osta.org/manifest/2.0"
       xmlns:mpv="http://ns.osta.org/mpv/1.0/"
       xmlns:mpv="http://ns.osta.org/mpv/music1.0/"
       xmlns:nmf="http://ns.osta.org/nmf/1.0/"
       xmlns:dc="http://purl.org/dc/elements/1.1/"
       xmlns:dcterms=http://purl.org/dc/terms/"
       xmlns:jpx="http://www.jpeg.org/jpx/">
  <nmf:Metadata>
    <file:ManifestProperties>
      <file:ProfileBag>
        <file:Profile>http://ns.osta.org/mpv/basic/is/1.0/</file:Profile>
        <file:Profile>http://ns.osta.org/mpv/basic/1.0/</file:Profile>
        <file:Profile>http://ns.osta.org/mpv/music/1.0/</file:Profile>
        <file:Profile>http://ns.osta.org/mpv/music/is/1.0/</file:Profile>
      </file:ProfileBag>
      <file2:AboutManifestMPVDocumentID>DOC001</file2:AboutManifestMPVDocumentID>
      <file2:WrittenBy>http://www.companyA.com/Device/Model1934/</file2:WrittenBy>
    </file:ManifestProperties>
  </nmf:Metadata>
  <mpv:AssetList>
    <mpv:Document mpv:id="DOC001">
      <mpv:LastURL mpv:filesystem="ISO9660-1">INDEX.PVM</mpv:LastURL>
      <mpv:LastURL mpv:filesystem="Unix">./index.pvm</mpv:LastURL>
      <mpv:LastURL>./album.pvm</mpv:LastURL>
```

```
      <nmf:Metadata>
        <dc:Properties>
          <dc:creator>Company A Application 1.3</dc:creator>
          <dc:description>this is some description</dc:description>
          <dc:language>en-US</dc:language>
          <dc:title>some title</dc:title>
        </dc:Properties>
        <dcterms:Properties>
          <dcterms:created>2002-03-12</dcterms:created>
        </dcterms:Properties>
      </nmf:Metadata>
    </mpv:Document>
    <mpv:Audio mpv:id="audio001">
      <mpv:LastURL mpv:filesystem="ISO9660-1">AUD001.MP3</mpv:LastURL>
      <mpv:LastURL mpv:filesystem="Unix">./aud01.mp3</mpv:LastURL>
      <mpv:LastURL>./aud01.mp3</mpv:LastURL>
      <nmf:Metadata>
        <dc:Properties>
          <dc:description>Some description</dc:description>
          <dc:format>audio/mpeg</dc:format>
          <dc:title>some title</dc:title>
        </dc:Properties>
        <mpvm:MusicProperties>
          <mpvm:PlayingTime>234.12</mpvm:PlayingTime>
          <mpvm:PrincipalArtist>Duke Ellington</mpvm:PrincipalArtist>
          <mpvm:Genre>Jazz</mpvm:Genre>
          <mpvm:Recorded>1955-11-17</mpvm:Recorded>
        </mpvm:MusicProperties>
      </nmf:Metadata>
    </mpv:Audio>

  </mpv:AssetList>
</file:Manifest>
```

Appendix: References

[DATETIME]

"Date and Time Formats", M. Wolf, C. Wicksteed. W3C Note 27 August 1998,
Available at: http://www.w3.org/TR/NOTE-datetime

[DC]

"Dublin Core Metadata Initiative", a Simple Content Description Model for Electronic Resources.
Available at http://purl.org/DC/

[DC-NMF]

"Dublin Core Normalized Metadata Format Profile Specification 1.0"; OSTA, 2002.
Available at http://www.osta.org/mpv/

[DCF-1999]

"Design rule for Camera File system, Version 1.0", JEIDA standard, English Version 1999.1.7, Japanese Electronic Industry Development Association (JEIDA).

[DIG35-2001]

"DIG35 Specification – Metadata for Digital Images, Version 1.1", June 18, 2001, International Imaging Industry Association (I3A) [recently formed by combining the Digital Imaging Group and PIMA]. http://www.i3a.org

[ISO8601]

"Data elements and interchange formats - Information interchange - Representation of dates and times", International Organization for Standardization, 1998.

[ISO10646]

""Information Technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane", ISO/IEC 10646-1:1993. This reference refers to a set of codepoints that may evolve as new characters are assigned to them. This reference therefore includes future amendments as long as they do not change character assignments up to and including the first five amendments to ISO/IEC 10646-1:1993. Also, this reference assumes that the character sets defined by ISO 10646 and Unicode remain character-by-character equivalent. This reference also includes future publications of other parts of 10646 (i.e., other than Part 1) that define characters in planes 1-16."

[JFIF]

"JPEG File Interchange Format, Version 1.02"; Eric Hamilton, September 1992.
Available at http://www.w3.org/Graphics/JPEG/jfif.txt

[MANIFEST]

"XML Manifest Specification 2.0"; OSTA
Available at http://www.osta.org/mpv/.[5]

[MD5]

"The MD5 Message-Digest Algorithm", RFC 1321, April 1992.
Available at http://www.ietf.org/rfc/rfc1321.txt. Further information and source code available at

5 Note that at the moment of writing of this document, the XML Manifest Specification 2.0 is not published yet. But it is guaranteed that after some time, it will be available from the given URI location.

http://userpages.umbc.edu/~mabzug1/cs/md5/md5.html

[MIME-2]

"RFC 2046: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types"; N. Freed, N. Borenstein, November 1996.
Available at ftp://ftp.isi.edu/in-notes/rfc2046.txt

[MIMETYPES-REG]

IANA official registry of MIME media types
Available at http://www.isi.edu/in-notes/iana/assignments/media-types/media-types

[MPVBasic]

"MPV – Basic Profile Specification", OSTA, 2002,
Available at http://www.osta.org/mpv/

[MPVCore]

"MPV Core Specification 1.0"; OSTA, 2002,.
Available at http://www.osta.org/mpv/

[MPVPres]

"MPV Presentation Profile Specification 1.0"; OSTA, 2002,.
Available at http://www.osta.org/mpv/

[NMF]

"Normalized Metadata Format Specification 1.0"; OSTA, 2002,.
Available at http://www.osta.org/mpv/

[PNG-MIME]

"Registration of new Media Type image/png"; Glenn Randers-Pehrson, Thomas Boutell, 27 July 1996.
Available at ftp://ftp.isi.edu/in-notes/iana/assignments/media-types/image/png

[PNG-REC]

"PNG (Portable Network Graphics) Specification Version 1.0"; Thomas Boutell (Ed.).
Available at http://www.w3.org/TR/REC-png

[QT]

"QuickTime Movie File Format Specification", May 1996.
Available at http://developer.apple.com/techpubs/quicktime/qtdevdocs/REF/refFileFormat96.htm

[QT-MIME]

"Registration of new MIME content-type/subtype"; Paul Lindner, 1993.
Available at http://www.isi.edu/in-notes/iana/assignments/media-types/video/quicktime

[RDFsyntax]

"Resource Description Framework (RDF) Model and Syntax Specification", Ora Lassila and Ralph R. Swick. W3C Recommendation 22 February 1999,
Available at http://www.w3.org/TR/REC-rdf-syntax/

[RDFschema]

"Resource Description Framework (RDF) Schema Specification", Dan Brickley and R.V. Guha.W3C Proposed Recommendation 03 March 1999,
Available at http://www.w3.org/TR/PR-rdf-schema/

[RFC1766]

"Tags for the Identification of Languages", H. Alvestrand, March 1995.
Available at ftp://ftp.isi.edu/in-notes/rfc1766.txt

[RFC2119]

“Key words for use in RFCs to Indicate Requirement Levels”, S. Bradner, IETF RFC 2119
http://www.ietf.org/rfc/rfc2119.txt

[URI]

"Uniform Resource Identifiers (URI): Generic Syntax", T. Berners-Lee, R. Fielding, L. Masinter, August 1998. Note that RFC 2396 updates [RFC1738] and [RFC1808].

[UCS-2]

16-bit encoding of ISO 10646, commonly known as the Unicode character set.

[UTF-8]

Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

[W3C-NSURI]

"URIs for W3C namespaces". Policy and administrative issue for W3C, Oct. 1999.
Available at http://www.w3.org/1999/10/nsuri

[XML10]

"Extensible Markup Language (XML) 1.0" T. Bray, J. Paoli and C.M. Sperberg-McQueen. W3C Recommendation 10 February 1998 ,
Available at http://www.w3.org/TR/REC-xml

[XML-NS]

"Namespaces in XML", Tim Bray, Dave Hollander, Andrew Layman. W3C Recommendation 14 January 1999,
Available at http://www.w3.org/TR/REC-xml-names

[XSCHEMA]

"XML Schema, XML Schema Part 1: Structures". W3C Working Draft, work in progress.
Available at http://www.w3.org/TR/xmlschema-1/

[XSL]

"Extensible Stylesheet Language (XSL) Specification", Stephen Deach. W3C Working Draft, work in progress.
Available at http://www.w3.org/TR/xsl/