Digital Music, Photo, and Video Collections

MPV Music Profile Specification

Working Draft
Revision 0.90
EXPIRES Sept. 1, 2003

23 July 2003

© 2002-2003 Optical Storage Technology Association
http://www.osta.org

IMPORTANT NOTICE
This document is a working draft for review by OSTA members and the general public. It is a draft document and will be updated, replaced, or obsoleted by other documents at any time and without notice. It is inappropriate to use OSTA Working Draft documents as reference materials, to cite them in other publications, or to refer to them as anything other than a “work in progress”.

NOT FOR DISTRIBUTION ON A PUBLIC WEBSITE
This document is available by sending a request to mpv-comment@list.osta.org
MAKING COMMENTS ON OSTA WORKING DRAFT SPECIFICATIONS

IMPORTANT NOTICES

(a) THIS DOCUMENT IS A DRAFT SPECIFICATION UNDER DEVELOPMENT BY THE OPTICAL STORAGE TECHNOLOGY ASSOCIATION (OSTA). THIS DRAFT SPECIFICATION IS MADE AVAILABLE FOR THE SOLE PURPOSE OF REVIEW AND COMMENT BY THE GENERAL PUBLIC AND IS NOT A FINAL SPECIFICATION OF OSTA. THIS DRAFT SPECIFICATION EXPIRES ON SEPTEMBER 1, 2003. OSTA MAY AT IT’S DISCRETION PUBLISH A FINAL SPECIFICATION AT A LATER TIME.

(b) ALL COMMUNICATIONS (CONTRIBUTIONS) TO OSTA BECOME THE PROPERTY OF OSTA AND MAY BE USED IN ANY WAY WITHOUT PERMISSION OF THE ORIGINATOR.

(c) TO BE ACCEPTED FOR CONSIDERATION BY OSTA, ALL PERSONS (CONTRIBUTORS) PROVIDING COMMENTS TO OSTA SHALL ABIDE BY THE FOLLOWING CONDITIONS.

   a. SENT BY EMAIL TO mpv-comment@list.osta.org.

   b. INCLUDE THE FOLLOWING INFORMATION ABOUT THE PERSON(S) PROVIDING THE CONTRIBUTIONS: FULL NAME, ORGANIZATION, EMAIL ADDRESS

   c. THE CONTRIBUTORS WILL GRANT A LICENSE, ON REASONABLE AND NONDISCRIMINATORY TERMS, ON A RECIPROCAL BASIS, UNDER PATENT CLAIMS ESSENTIAL TO IMPLEMENT THIS SPECIFICATION. FURTHER INFORMATION MAY BE OBTAINED FROM OSTA.

   d. THE CONTRIBUTORS SHALL DISCLOSE IN THE COMMENTS ANY PATENT CLAIMS OF WHICH THEY ARE AWARE THAT MAY BE OR ARE ESSENTIAL TO IMPLEMENT ANY ASPECT OF THE COMMENTS.

   e. ALL CONTRIBUTIONS ARE OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED, EXCEPT TO THE EXTENT OF KNOWING FALSEITY IN ANY STATEMENT MADE ABOVE. ANY USE OF ANY SPECIFICATION INCORPORATING THIS CONTRIBUTION IN WHOLE OR IN PART SHALL BE MADE ENTIRELY AT THE USER’S OWN RISK, AND THE CONTRIBUTOR SHALL HAVE NO LIABILITY WHATSOEVER TO ANY USER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER DIRECTLY OR INDIRECTLY ARISING FROM SUCH USE, EXCEPT AS A RESULT OF ANY KNOWING FALSEITY IN ANY STATEMENT MADE ABOVE. ANY LICENSE BY OSTA OF A FINAL SPECIFICATION INCORPORATING THIS CONTRIBUTION SHALL INCLUDE A DISCLAIMER IN SUBSTANTIALLY THE FORM SET FORTH IN THIS PARAGRAPH.
ABSTRACT

The Music Profile specification defines metadata and practices for processing and playback of collections of digital music collections stored on an optical disc and other storage media such as memory cards and computer harddrives or exchanged via internet protocols.

COPYRIGHT NOTICE

Copyright 2002-2003 Optical Storage Technology Association, Inc.

RELEASE HISTORY

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.21</td>
<td>2 December 2002</td>
<td>First release to OSTA membership.</td>
</tr>
<tr>
<td>0.30</td>
<td>7 January 2003</td>
<td>Updated logo, contact info, copyright. No other changes.</td>
</tr>
<tr>
<td>0.40</td>
<td>18 January 2003</td>
<td>Completed draft spec.</td>
</tr>
<tr>
<td>0.50</td>
<td>20 January 2003</td>
<td></td>
</tr>
<tr>
<td>0.70</td>
<td>22 January 2003</td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>31 January 2003</td>
<td></td>
</tr>
<tr>
<td>0.76</td>
<td>24 March 2003</td>
<td>Reflects inputs from OSTA Multiread Committee review</td>
</tr>
<tr>
<td>0.80</td>
<td>14 July 2003</td>
<td>Acting on suggested changes from many parties. Added ID3v1 compatibility; changed to Music Manifest file type; added new media types; Changed OrigIndex to TrackNumber. Added SetNumber, NumSets, NumTracks. Added additional music metadata properties. Added AudioWithStills asset.</td>
</tr>
<tr>
<td>0.90</td>
<td>24 July 2003</td>
<td>Released for public comment.</td>
</tr>
</tbody>
</table>
LICENSING

IMPORTANT NOTICES

(a) THIS DOCUMENT IS A DRAFT SPECIFICATION UNDER DEVELOPMENT BY THE OPTICAL STORAGE TECHNOLOGY ASSOCIATION (OSTA). THIS DRAFT SPECIFICATION IS MADE AVAILABLE FOR THE SOLE PURPOSE OF REVIEW AND COMMENT BY THE GENERAL PUBLIC AND IS NOT A FINAL SPECIFICATION OF OSTA. IT WILL BE UPDATED, REPLACED OR OBSELETED BY OTHER DOCUMENTS AT ANY TIME AND WITHOUT NOTICE. IT IS NOT APPROPRIATE TO USE OSTA DRAFT SPECIFICATIONS AS REFERENCE MATERIALS, TO CITE THEM IN OTHER PUBLICATIONS, OR TO REFER TO THEM AS ANYTHING OTHER THAN A "WORK IN PROGRESS".

(b) THIS DOCUMENT IS AN AUTHORIZED AND APPROVED PUBLICATION OF THE OPTICAL STORAGE TECHNOLOGY ASSOCIATION (OSTA). THE SPECIFICATIONS CONTAINED HEREIN ARE THE EXCLUSIVE PROPERTY OF OSTA BUT MAY BE REFERRED TO AND UTILIZED BY THE GENERAL PUBLIC FOR ANY LEGITIMATE PURPOSE, PARTICULARLY IN THE DESIGN AND DEVELOPMENT OF WRITABLE OPTICAL SYSTEMS AND SUBSYSTEMS. 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.

(c) COMPLIANCE WITH THIS DOCUMENT MAY REQUIRE USE OF ONE OR MORE FEATURES COVERED BY THE PATENT RIGHTS OF AN OSTA MEMBER, ASSOCIATE 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 OR ASSOCIATE 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 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) ONE OR MORE PATENT HOLDERS HAVE FILED STATEMENTS OF WILLINGNESS TO GRANT A LICENSE, ON REASONABLE AND NONDISCRIMINATORY TERMS, ON A RECIPROCAL BASIS, UNDER PATENT CLAIMS ESSENTIAL TO IMPLEMENT THIS SPECIFICATION. FURTHER INFORMATION MAY BE OBTAINED FROM OSTA.

(e) 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.

(f) THIS DOCUMENT IS A SPECIFICATION ADOPTED BY OSTA. 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 IS PROVIDED "AS IS".

(g) THE NAMES MusicPhotoVideo AND MPV AND THE MPV LOGO ARE TRADEMARKS OF OPTICAL STORAGE TECHNOLOGY ASSOCIATION, INC. ALL OTHER TRADEMARKS ARE THE PROPERTY OF THEIR RESPECTIVE OWNERS. THE MusicPhotoVideo and MPV NAMES AND MPV LOGO MAY NOT BE USED EXCEPT FOR JOURNALISTIC PURPOSES WITHOUT AN EXPLICIT LICENSE FROM OSTA.
Contents

Chapter 1: Introduction ................................................................. 5
  1.1 Executive Summary ........................................................... 7
  1.2 Terms of Use ..................................................................... 9
Chapter 2: MPV Music Profile 1.0 .................................................. 10
  2.1 MPV Music Profile Introduction ....................................... 10
  2.2 Formalities For Use of the MPV Music Profile ................. 11
Chapter 3: MPV Music Schema Introduction ................................ 13
  3.1 Introduction ..................................................................... 13
  3.2 Examples ........................................................................ 13
    3.2.1 Namespaces and Profiles ....................................... 13
    3.2.2 Simple Example .................................................... 14
    3.2.3 Rich Example ....................................................... 15
  3.3 Use of Existing MPV Specifications ................................. 18
  3.4 MPV Music Profile Metadata Introduction ....................... 18
  3.5 Use of Dublin Core Metadata ............................................ 21
Chapter 4: MPV Music Profile – Using MPV Playlists ................. 23
  4.1 MPV Music Playlists ......................................................... 23
    4.1.1 Number of Playlists per MPV File ....................... 23
    4.1.2 Metadata Usage .................................................... 24
    4.1.3 Background Usage ............................................... 24
    4.1.4 Foreground Usage ............................................... 24
    4.1.5 Related and Rendition Assets .............................. 25
  4.2 Groups of Assets in Playlists ............................................ 25
  4.3 Linked Playlists ............................................................... 27
  4.4 Links to Foreign Playlists ................................................. 29
  4.5 Default Playlists ............................................................. 31
  4.6 Dynamic Playlists ........................................................... 31
    4.6.1 Specifying Dynamic Playlists .............................. 32
    4.6.2 Playlist Query Language ..................................... 32
  4.7 Best Practices for Generating Playlists ......................... 32
    4.7.1 Top-Level Playlist ................................................. 32
    4.7.2 “All Music” Playlist ............................................. 33
    4.7.3 Local mpv:AssetList ........................................... 33
    4.7.4 Centralized mpv:AssetList .................................. 33
Chapter 5: MPV Music Schemas In Detail .............................. 34
  5.1 Multiple Renditions of a Music Asset ............................ 34
  5.2 Artwork for an Asset ...................................................... 36
  5.3 Asset-related Content .................................................... 37
  5.4 Music Identification ....................................................... 37
Chapter 1: Introduction

1.1 Executive Summary

1.1.1 The Business Case

A common use model for digital music is emerging in the digital home: the PC is the central repository of large collections of digital music (commonly, thousands of files) and the playback, or enjoyment, of the music happens on a myriad of consumer electronics devices throughout the home and beyond -- devices such as CD/DVD players, portable music players, car stereos and wireless network receivers that stream the music from the PC to home entertainment systems.

The software media players on the PC have added many exciting management features. For example, consumers can easily create multiple playlists to quickly playback their favorite music or music that suits a particular mood or need. However, many of these playlists are in a proprietary format and not universally recognized. Another management feature of today’s media players is to recognize the performer of a song file on the PC and automatically add the metadata to the file. This metadata – artist, song title, album, year, genre, etc. -- is typically embedded into each file. When 2000+ songs are burned onto a recordable DVD which is then inserted into a DVD player, it can take the consumer electronics device up to 2 minutes to initialize as it reads each individual file to find the metadata to create its navigation scheme. This metadata is once again lost when the user “burns” an Audio CD (Redbook) from these files – a common problem for online subscription services since they then lose track of the file purchased on their service.

To date, there are three ways to move the music from its central repository on the PC to these devices in order to play them back:

1. Tethered connection (typically USB or IEEE1394 /Firewire) for many portable devices
2. Recordable media such as blank CDs or DVDs for car stereos or DVD players
3. Home Network connection (wired or wireless) for newer digital media receivers

In the first two (and most common) cases, the consumer copies and therefore duplicates their music onto the CD, DVD or hard disk to play it back. After a period of time, the user has accumulated multiple copies of the same file spread across many discs and hard drives complicating overall management. In the third case, the networked device is often connected to a television which allows for a sophisticated interface and applications. Examples of such applications are the creation of playlists or the display of additional related visual data such as album covers or artist photos. As such devices proliferate in the home, compatibility of playlists between the various makes and models of devices becomes paramount.

All this will become even more complex as other devices vye to become the central repository for music. The Personal Video Recorders (PVRs) already have large storage capacity that allow consumers to “rip” their AudioCDs onto the hard disk drive. PVRs are also connected to services that will soon be offering subscription or download
purchase of music. This means that some songs may be on the PC while others are only on the PVR. The consumer will have a difficult time finding files spread out over two or more unconnected devices.

In time, consumers will have increasing difficulties managing their collections. This calls for a common, accepted way to represent digital music collections and manage the playlists and metadata. This is the overarching goal of the MPV Music Profile.

Here are some exciting ways in which MPV could create a better music experience and improve business processes in the digital music world:

- When consumers use an MPV-enabled disc recording application to make a compilation CD, the application will also record the MPV information (song identification, playlists and metadata) in a single file to the disc. Consumer playback devices could then quickly read that one file and determine the contents of the disc to display an easy-to-use navigation system enabling users to search or sort by genre, artist, album or song.
- The new wireless receivers in home networks that stream music from the personal computer use Universal Plug and Play (UPnP), an XML specification, to communicate. MPV, also XML-based, can work gracefully with UPnP to standardize the playback of collections and playlists from and to all devices on the home network.
- Music labels could add MPV information along with cover art, artist photos and lyrics to pre-recorded audio CDs that could be played back on consumer electronics devices. For example, when listening to an audio CD on a DVD player, the consumer could view a photo slideshow of the artist on the TV screen. The music industry often adds exciting content to Audio CDs that only plays back on PCs with added software. With MPV, it would be possible to access this content on consumer electronics devices.
- MPV could also be used by online music subscription services to track songs the user burns onto a recordable disc. The MPV information would be in the data portion of a mixed mode disc.

1.1.2 MusicPhotoVideo Specification Overview

MPV (MusicPhotoVideo) provides multimedia playlists and access to their associated metadata. MPV is an open specification that 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.

Applications and devices and users that use MPV benefit even when they only interact with music and audio in basic ways; such as personal music collections that can be burned on CDs by many software applications.

MPV uses a text-based format that is easily understood and also easy to produce and consume programmatically in firmware or computer software. MPV does not tackle a large number of problems at once – instead, it focuses on a few key problems that it solves with simple but robust approaches. Where possible and practical, it supports use of established specifications and standards.

The development and promotion of MPV is sponsored by the Optical Storage Technology Association (OSTA). The specification development and promotion process is open to all members; all organizations and individuals are welcomed as members. The association includes over 50 member companies from all over the world that produce products that collectively represent a majority marketshare in mainstream recordable optical storage categories.

MPV is not only a specification. It also includes a compliance test suite and processes, compliance testing materials, a logo program for compliant products, and a website. These materials and procedures are made available and administered by OSTA at a modest cost. OSTA charges no royalty for use of the specification or logo. In addition, sample open-source code implementations of key steps in processing MPV content are being contributed by interested parties.

The specification is being developed in phases and results in "profiles". Each profile in MPV defines only those formats and practices that are necessary for the key tasks targeted by the profile. A number of candidate profiles for development have been identified, including:
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 slideshows and interactive browsing of photo-video content. The Presentation Profile is also used by the Music Profile to as a music playlist.

MPV technology has three central components: Collections, Metadata, and Identification. Each of these make reference in various ways to data files containing the music, photo, or video content. This information may be augmented by information from various profiles. For example, the Presentation profile provides information that may be used by player applications and devices to provide an attractive playback user experience.

### 1.2 Terms of Use

This section of the specification is descriptive and not intended to be complete nor definitive. Please refer to the definitive statement of licensing terms at the beginning of the MPV specification document for a precise and legal description.

The MPV specification is developed using an open process. The resulting specification is available from OSTA. No royalty is charged by OSTA for use of the specification. The overall desire is to develop a specification that is not subject to separate licensing requirements or royalty. During the development process, the expectation is that all participants contribute their efforts and intellectual property without any expectation or requirement for compensation. However, OSTA does not warrant that the specification is not or will not be subject to such claims by other parties.

MPV is not only a specification. It also includes a compliance test suite and processes, compliance testing materials, a logo program for compliant products, and a website. These materials and procedures are made available and administered by OSTA at a modest cost. OSTA charges no royalty for use of the specification. In addition, some sample open-source code implementations of key steps in processing MPV content may be contributed by interested parties.
Chapter 2: MPV Music Profile 1.0

The MPV Music Profile allows users, via applications and devices, to create and playback collections of music organized into albums / playlists. The MPV Music Profile extends the existing MPV Core specification and Basic and Presentation Profiles by augmenting this framework with additional metadata and practices specific to music.

A user may organize their music content into collections and burn it on a recordable CD or DVD. When the music collections are represented on the disc using MPV files that implement the MPV Music Profile, then a playback application or device can quickly start playback as soon as the disc is inserted and allow the user to easily navigate and playback the music, equally well and quickly regardless of whether the disc has 15, 150, or 1500 songs on it. Of course, in addition to basic music playback, an application or device can show basic music information like title, artist and genre – this may be retrieved from the music files themselves or from the MPV collection. Additional content may also be part of the collection and is available to be shown by the playback application or device, such as artwork related to the music, lyrics, and even music videos of the songs.

The Music Profile provides a basic set of metadata which represents data and conventions used by software applications that create and play compressed audio music on PCs or consumer electronics devices and music publishers of music CDs.

2.1 MPV Music Profile Introduction

The MPV Music Profile 1.0 supports the following key tasks: defining collections of music, organizing music into albums and playlists, listening sequentially or shuffled to an album / playlist, and interactively browsing single or multiple album / playlists of music.

The music metadata that may be represented using the MPV Music Profile includes the following:

**Music Asset (“Song”, “Track”):** Filename, Title, Principal artist, Album title, Genre, Playing time, Date recorded, Track number, Num Tracks, Set number, Num sets, Artwork, Music videos, Performed by, Music by, Lyrics by, Arranged by, More info URL, Encoded bitrate, Lyrics, Rights, Identifier, Description, Format.

The MPV Music Profile can also organize music content in useful and novel ways. For example, a music asset may have multiple representations, such as multiple bitrate encodings, multiple format encodings (which may enhance compatibility with devices), and multiple representations, such as audio-only, video-with-audio, and song artwork.

**Album (“Playlist”) of Music:** Title, Principal Artist, Description, Identifier, Artwork, Music Entries

The capabilities of the MPV Music Profile allows discs to be produced that have variable user experiences depending on the type of device used to play them. For example, a low-cost CD player could just play MP3 music...
and display information on a 4 line LCD display. A capable DVD player could play music videos and display music information on a multi-line graphical display along with artwork and lyrics.

MPV also allows music to be organized into hierarchical playlists, allowing users to navigate among playlists that may be both pre-generated or created on-the-fly by the playback application.

### 2.2 Formalities For Use of the MPV Music Profile

The mechanism that MPV uses to add capabilities to the Core specification is the Profile. MPV Core sets out specific formalities to follow when a MPV Profile is used -- an 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.

#### PROFILE COMPONENTS

The MPV Music Profile 1.0 makes use of two other specifications:

- MPV Core Specification 1.0
- MPV Presentation Profile Specification 1.0

The MPV Music Profile 1.0 includes the schema and practices detailed by this document.

#### COMPATIBILITY

The MPV Music Profile 1.0 is an extension of the MPV Core Specification 1.0 and is fully compatible with the MPV framework it establishes. Thus MPV files that implement the MPV Music Profile should be usable in basic ways by MPV-aware applications and devices not focused on music playback. This means, for example, that a MPV playback application or device can read and playback MPV music collections even if it doesn’t understand the MPV Music Profile; however, the music-specific information will be ignored and the playback experience will be less full-featured than in a MPV Music Profile player.

#### SCHEMANAMESPACE

To use the MPV Music Profile, this information must be present in the namespace declarations in the MPV file:

<table>
<thead>
<tr>
<th>Schema</th>
<th>Namespace Identifier</th>
<th>Schema Location</th>
<th>Conventional Namespace Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Music Profile</td>
<td><a href="http://ns.osta.org/mpv/music/1.0/">http://ns.osta.org/mpv/music/1.0/</a></td>
<td>lax/profiles/music/profile.xsd</td>
<td>mpvm:</td>
</tr>
</tbody>
</table>

The schema location may be specified optionally. Multiple schema variations may exist depending on the degree of validation desired by the developer. Typical variations include “lax”, “strict”, and “fixed”. These schema will all implement the grammar of the MPV Music Profile but will vary in the degree of flexibility and conformance requirements that they embody.

#### PROFILE IDENTIFIER

This information must be present in the Profile section of the MPV Manifest.

| Music Profile Name | http://ns.osta.org/mpv/music/1.0/ |
EXAMPLE

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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>
    <ManifestProperties xmlns="http://ns.osta.org/manifest/1.0/">
      <ProfileBag>
        <Profile>http://ns.osta.org/mpv/basic/1.0/</Profile>
        <Profile>http://ns.osta.org/mpv/presentation/1.0/</Profile>
        <Profile>http://ns.osta.org/mpv/music/1.0/</Profile>
      </ProfileBag>
    </ManifestProperties>
  </nmf:Metadata>
  ...
</file:Manifest>
```
Chapter 3: MPV Music Schema
Introduction

3.1 Introduction

The MPV Music Profile makes use of the existing MPV Core specification and Basic and Presentation Profiles for creating collections of music and organizing them into albums / playlists. The MPV Music Profile augments this framework with additional metadata and practices specific to music.

The Music Profile provides a basic set of metadata which represents data and conventions used by software applications that create and play audio music on PCs or consumer electronics devices and music publishers of music CDs. The music metadata that may be represented using the MPV Music Profile includes the following:

Music Asset ("Song", "Track"): Asset Filename, Title, Principal artist, Album title, Genre, Playing time, Year recorded, Original order, Artwork, Music video, Performed by, Music by, Lyrics by, Arranged by, More info, Average encoded bitrate, Lyrics, Rights, Identifier, Description, Format,

Album ("Playlist") of Music: Title, Principal Artist, Description, Identifier, Artwork, Music Entries

3.2 Examples

MPV Music playlists can range from simple to sophisticated, depending on the amount of available information and the ability of the creating application or device. Playback applications and devices determine the extent to which they use available information and the presentation of that information.

3.2.1 Namespaces and Profiles

All MPV files begin with a preamble that declares the XML namespaces and profiles used by the file. The `xmlns:xyz="namespace identifier"` sequence assigns a shortcut prefix (xyz) to represent the unique namespace identifier within the file. Use of namespaces allows the same element name to be used from different schema without ambiguity. For example, `<foo:Element>` and `<bar:Element>` are different if the namespace identifiers for each prefix are different and are the same if the namespace identifiers are the same.

A typical preamble:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest xmlns:file="http://ns.osta.org/manifest/1.0/
xmlns:mpv="http://ns.osta.org/mpv/1.0/"
```
The Music profile in the manifest properties. As a best practice, the Basic profile also SHOULD be listed in the manifest. If an `<mpvp:Album>` element is provided, then the Presentation profile SHOULD be listed in the manifest. Adding the Basic and Presentation profile entries allow players that only understand those profiles to provide at least simple playback of a MPV file implementing the Music Profile.

### 3.2.2 Simple Example

This example MPV Music Profile file has 6 songs with only file location info for each item. There is no `<mpvp:Album>` playlist, so the sequence is the order of appearance in the `<mpv:AssetList>`. This example is the simplest form of using MPV for music playlists. Note that the file location of each song is provided in the two filesystems that typically occur on a CD, which are Joliet and ISO9660-1.

Even with this very simple usage, this MPV playlist adds value to the user’s playback experience because the order of music playback is specified explicitly and is different from the sort order of the music by filename or file date.
3.2.3 Rich Example

In contrast to the previous example, this example of a MPV Music Profile file has much more information. In this case, two songs are specified along with a lot of information about the music including album artwork and music videos for the first song, and also a playlist (the mpvp:Album element) is provided that specified album/playlist-level information.

Careful reading of the contents of the mpv:AssetList in this example will show that not only the music songs but also still image and video assets are listed plus statements that relate the assets to each other. Not all these assets are considered “primary”, in other words, the user doesn’t want to interact with all assets equally. Primary assets are the ones that match the user’s idea of what the primary content is, such as a set of music songs. The mpvp:Album element is used to identify the primary assets, the sequence in which they should be presented, and other presentation information.

This example also illustrates how MPV Music Profile can be applied to a “hybrid” disc, such as a disc with that is both a DVD-Video disc and also contains MPV Music playlists and MP3 music of the songs. When played in a DVD-Video player, the user may enjoy watching the DVD-Video content, such as a music performance. In this case, no MPV Music information is used, just DVD-Video content and navigation. However, when played in a car stereo, only the MPV Music information is used and the player plays the MP3 music tracks that are also on the disc.

Some players will support both DVD-Video and MPV Music-based playback. In that case, for example, the MPV Music playback application may choose to allow the user to playback the associated music video for a track. In this example, the music video specified in the MPV playlist is actually the same music video played in DVD-Video mode, but it is accessed in a different way.
<ManifestProperties xmlns="http://ns.osta.org/manifest/1.0/">
  <ProfileBag>
    <Profile>http://ns.osta.org/mpv/basic/1.0/</Profile>
    <Profile>http://ns.osta.org/mpv/presentation/1.0/</Profile>
    <Profile>http://ns.osta.org/mpv/music/1.0/</Profile>
  </ProfileBag>
</ManifestProperties>

<mpvp:Album> <!-- This defines an album / playlist presentation of the assets -->
  <nmf:Metadata> <!-- Album / playlist–level information -->
    <dc:Properties>
      <dc:description>14 swing classics re-recorded in the '50s by the original artists for great sound with all the integrity and excitement of the original performances.</dc:description>
      <dc:identifier>7243 5 21223 2 5 Capitol Jazz</dc:identifier>
      <dc:rights>(P) and (C) 1999 Capitol Records, Inc. All rights reserved.</dc:rights>
      <dc:title>Music by Album and Track</dc:title>
    </dc:Properties>
    <mpvm:MusicProperties>
      <mpvm:AlbumTitle>Great SWING CLASSICS in HI-FI</mpvm:AlbumTitle>
      <mpvm:Genre>Jazz</mpvm:Genre>
    </mpvm:MusicProperties>
  </nmf:Metadata>
  <mpvp:Foreground> <!-- music playback sequence -->
    <mpv:AudioRef mpv:idRef="01-GREAT-SWING-CLASSICS.MP3-20021202031833-a"/>
    <mpv:AudioRef mpv:idRef="02-GREAT-SWING-CLASSICS.MP3-20021202031833-a"/>
  </mpvp:Foreground>
</mpvp:Album>

<mpv:AssetList> <!-- This is the per-asset info -->
  <mpv:Audio mpv:id="01-GREAT-SWING-CLASSICS.MP3-20021202031833-a">
    <mpv:LastURL mpv:filesystem="Joliet">Benny%20Goodman%20And%20His%20Orchestra%20-%20Jumpin%20At%20The%20Woodside.mp3</mpv:LastURL>
    <mpv:LastURL mpv:filesystem="ISO9660-1">BENNY_GO.MP3</mpv:LastURL>
  </mpv:Audio>
  <nmf:Metadata>
    <dc:Properties>
      <dc:creator>Benny Goodman and his Orchestra</dc:creator>
      <dc:description/>
      <dc:format>audio/mpeg</dc:format>
      <dc:identifier/>
      <dc:title>Jumpin' At The Woodside</dc:title>
    </dc:Properties>
    <mpvm:MusicProperties>
      <mpvm:AlbumTitle>Great SWING CLASSICS in HI-FI</mpvm:AlbumTitle>
      <mpvm:ArrangedBy>Count Basie; Jimmy Mundy</mpvm:ArrangedBy>
      <mpvm:Genre>Jazz</mpvm:Genre>
      <mpvm:MusicBy>Count Basie</mpvm:MusicBy>
      <mpvm:NumTracks>14</mpvm:NumTracks>
      <mpvm:PlayingTime>208.6</mpvm:PlayingTime>
      <mpvm:PrincipalArtist>Benny Goodman</mpvm:PrincipalArtist>
      <mpvm:Recorded>1954-11-09</mpvm:Recorded>
      <mpvm:TrackNumber>1</mpvm:TrackNumber>
    </mpvm:MusicProperties>
  </nmf:Metadata>
</mpv:AssetList>
Filenames of assets are specified using the `mpv:LastURL` element. Pathnames can be relative or absolute; relative names begin relative to the location of the MPV file. Pathnames are to be specified using URL-compliant syntax. This includes translation of special characters like the space (" ") into equivalent representations like “%20” and use of prefixes like `file:///` to introduce absolute pathnames to local files. Multiple pathnames may be specified for any given asset; they are interpreted as alternate paths to the same set of bits. A processing application tries them sequentially to try and locate the asset.

The term “LastURL” is used to emphasize that it’s value is a URL to the last-known location of the file; because media files may be moved or renamed independently of the MPV file, it is possible that the media file has moved and must be searched for. The `mpv:InstanceID` and `mpv:ContentID` elements, if provided, are identifiers that can be used to find files that cannot be located by any of the `LastURL` entries.

### 3.3 Use of Existing MPV Specifications

The MPV Music Profile uses MPV in a manner consistent with these the existing MPV Core specification and the MPV Basic Profile and Presentation Profile specifications. For metadata, it incorporates the MPV Dublin Core NMF specification for those properties that can be represented in that manner.

### 3.4 MPV Music Profile Metadata Introduction

The MPV Core specification already supports an `mpv:Audio` asset type. The MPV Presentation Profile specification describes how to create playlists ("mpvp:Album") of assets, such as music and images. To this framework, the MPV Music Profile adds extensive metadata specifically about music; the existing framework continues to be used in a manner fully consistent with existing specifications.

<table>
<thead>
<tr>
<th>Metadata</th>
<th>MPV Music Profile</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Music Asset</td>
<td>Subelements of the <code>mpv:Audio</code> asset</td>
<td></td>
</tr>
<tr>
<td>“Song”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“Track”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pathname</td>
<td><code>mpv:LastURL</code></td>
<td>one or more pathnames that should resolve to the music file. Each pathname is specific to a filesystem, so there can be different names for each filesystem on a CD.</td>
</tr>
<tr>
<td>Format</td>
<td>`nmf:Metadata</td>
<td>dc:Properties</td>
</tr>
<tr>
<td>Title</td>
<td>`nmf:Metadata</td>
<td>dc:Properties</td>
</tr>
<tr>
<td>Album Title</td>
<td>`nmf:Metadata</td>
<td>mpvm:MusicProperties</td>
</tr>
<tr>
<td>Property</td>
<td>Namespace</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Performed by</td>
<td>nmf:Metadata</td>
<td>dc:Properties</td>
</tr>
<tr>
<td>Principal artist</td>
<td>nmf:Metadata</td>
<td>mpvm:MusicProperties</td>
</tr>
<tr>
<td>Band, Orchestra,</td>
<td>nmf:Metadata</td>
<td>dc:Properties</td>
</tr>
<tr>
<td>Accompaniment</td>
<td>Music by</td>
<td>Composer name(s) in free form plain text.</td>
</tr>
<tr>
<td></td>
<td>Lyrics by</td>
<td>Lyricist name(s) in free form plain text.</td>
</tr>
<tr>
<td>Arranged by</td>
<td>Produced By</td>
<td>Arranger name(s) in free form plain text.</td>
</tr>
<tr>
<td></td>
<td>Genre</td>
<td>Predefined set of genres can be extended simply by using a new name.</td>
</tr>
<tr>
<td>Date Recorded</td>
<td>Identifier</td>
<td>Recorded date-time, e.g. 2002-10-03T21:07:00Z.</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Commentary, message, description</td>
</tr>
<tr>
<td></td>
<td>Keywords</td>
<td>Semicolon separated list of plain text keywords or phrases for search. Semi-colon can be used by escaping with backslash, e.g. \;</td>
</tr>
<tr>
<td>Rendition using another</td>
<td>Rendition</td>
<td>Refers to an alternate encoding of the asset.</td>
</tr>
<tr>
<td>encoder</td>
<td>mpv:Rendition mpv:renditionType=&quot;alt&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mpv:AudioRef</td>
<td></td>
</tr>
<tr>
<td>Rendition with another</td>
<td>Rendition</td>
<td>Refers to another encoding of the asset at a different bitrate. Use mpvm:EncodedBitrate or interrogate the file for more information.</td>
</tr>
<tr>
<td>bitrate</td>
<td>mpv:Rendition mpv:renditionType=&quot;subsampled&quot;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>mpv:AudioRef</td>
<td></td>
</tr>
<tr>
<td>Encoded Bitrate</td>
<td>nmf:Metadata</td>
<td>mpvm:MusicProperties</td>
</tr>
<tr>
<td>Asset artwork</td>
<td>mpv:Related mpv:relationship=&quot;urn:osta-&quot;</td>
<td>Refers to an image or</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Music video</td>
<td>mpv:Related relationship=&quot;urn:osta-org:mpv:music:video&quot;</td>
<td>mpv:VideoRef</td>
</tr>
<tr>
<td>Online Info</td>
<td>nmf:Metadata</td>
<td>mpvm:MusicProperties</td>
</tr>
<tr>
<td>Playing Time</td>
<td>nmf:Metadata</td>
<td>mpvm:MusicProperties</td>
</tr>
<tr>
<td>Track Number</td>
<td>nmf:Metadata</td>
<td>mpvm:MusicProperties</td>
</tr>
<tr>
<td>Number Tracks</td>
<td>nmf:Metadata</td>
<td>mpvm:MusicProperties</td>
</tr>
<tr>
<td>Set Number in Collection</td>
<td>nmf:Metadata</td>
<td>mpvm:MusicProperties</td>
</tr>
<tr>
<td>Number Sets in Collection</td>
<td>nmf:Metadata</td>
<td>mpvm:MusicProperties</td>
</tr>
<tr>
<td>PlayCount</td>
<td>nmf:Metadata</td>
<td>mpvm:MusicProperties</td>
</tr>
<tr>
<td>Lyrics</td>
<td>nmf:Metadata</td>
<td>mpvm:MusicProperties</td>
</tr>
<tr>
<td>Mood</td>
<td>nmf:Metadata</td>
<td>mpvm:MusicProperties</td>
</tr>
<tr>
<td>Tempo</td>
<td>nmf:Metadata</td>
<td>mpvm:MusicProperties</td>
</tr>
<tr>
<td>Situation</td>
<td>nmf:Metadata</td>
<td>mpvm:MusicProperties</td>
</tr>
<tr>
<td>Medium</td>
<td>nmf:Metadata</td>
<td>dcterms:Properties</td>
</tr>
<tr>
<td>Original file and filename</td>
<td>mpv:Related relationship=&quot;copyOf&quot;</td>
<td>mpv:AudioRef</td>
</tr>
</tbody>
</table>
3.5 Use of Dublin Core Metadata

To promote interoperability, MPV makes use of the Dublin Core metadata [DC-NMF] to represent essential metadata across all types of assets. Thus, the dc:title element is used to specify the title of a music asset just the same as the title of an image or video asset.

In the previous section, the Dublin Core metadata elements were mixed into the overall set of music metadata properties and assets. To clarify usage of the DC metadata, this section extracts just the Dublin Core elements from the previous section and groups them together for convenience.
<table>
<thead>
<tr>
<th>Performed by</th>
<th>nmf:Metadata</th>
<th>dc:Properties</th>
<th>dc:creator</th>
<th>Performer Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>Band, Orchestra, Accompaniment</td>
<td>nmf:Metadata</td>
<td>dc:Properties</td>
<td>dc:contributor</td>
<td>Band, orchestra, accompaniment</td>
</tr>
<tr>
<td>Description</td>
<td>nmf:Metadata</td>
<td>dc:Properties</td>
<td>dc:description</td>
<td>Commentary, message, description</td>
</tr>
<tr>
<td>Format</td>
<td>nmf:Metadata</td>
<td>dc:Properties</td>
<td>dc:format</td>
<td>use MPV-provided list of well-known MIME types</td>
</tr>
<tr>
<td>Identifier</td>
<td>nmf:Metadata</td>
<td>dc:Properties</td>
<td>dc:identifier</td>
<td>Catalog number, UPC, etc. A text string, not classified in any way and not guaranteed unique.</td>
</tr>
</tbody>
</table>
| Keywords                     | nmf:Metadata|dc:Properties|dc:subject         | Semicolon separated list of plain text keywords or phrases for search. Semicolon can be used by escaping with backslash, e.g. \\
| Publisher                    | nmf:Metadata|dc:Properties|dc:publisher       | Free-form plain text |
| Rights                       | nmf:Metadata|dc:Properties|dc:rights          | Free-form plain text |
| Title                        | nmf:Metadata|dc:Properties|dc:title           | Free-form plain text |
| Created Date                 | nmf:Metadata|dcterms:Properties|dcterms:created   | date-time music performed or recorded |
| Medium                       | nmf:Metadata|dcterms:Properties|dcterms:medium    | Original music release medium conformant to [ID3v240] |

**Album (“Playlist”) of Music**

<table>
<thead>
<tr>
<th>mpvp:Album subelements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performed By</td>
</tr>
<tr>
<td>Band, Orchestra, Accompaniment</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Identifier</td>
</tr>
<tr>
<td>Publisher</td>
</tr>
<tr>
<td>Rights</td>
</tr>
<tr>
<td>Title</td>
</tr>
</tbody>
</table>
Chapter 4: MPV Music Profile – Using MPV Playlists

4.1 MPV Music Playlists

MPV playlists are a central concept in MPV that provide for user-friendly organization and navigation of the music, photo, and video (and other) content on a storage media. A MPV playlist identifies the primary assets to be presented to the user, such as a set of music songs. The playlist also specifies the sequence in which assets should be presented and other presentation characteristics.

MPV playlists are implemented according to the MPV Presentation Profile specification [MPV-Pres]. This MPV Music Profile document specifies the details of this usage. The basic structure of an MPV playlist uses the <mpvp:Album> element, which may contain metadata, foreground and background content, and related and rendition assets.

```xml
<mpvp:Album>
  <mpv:Metadata> ... </mpv:Metadata>
  <nmf:Metadata> ... </nmf:Metadata>
  <mpvp:Background> ... </mpvp:Background>
  <mpvp:Foreground> ... </mpvp:Foreground>
  <mpv:Rendition> ... </mpv:Rendition>
  <mpv:Related> ... </mpv:Related>
</mpvp:Album>
```

MPV is focused on interoperability of content produced and consumed on both PCs and consumer electronics devices. Unlike CE-focused solutions, it is expected that users will organize media files into folders/directories of their own choosing and give files long filenames. This means that it isn’t possible to specify content location and playback order simply by requiring specific directory and filenames. Instead, MPV provides an approach that allows the content to be located anywhere and with any name.

In addition, many related media assets may go onto a storage media such as CD that are used to enhance access and playback performance and to provide enhanced playback and printing experiences. The MPV playlist allows this content to be managed to present the user a simple high-level interaction with their content.

4.1.1 Number of Playlists per MPV File

A MPV file conforming to the MPV Music Profile 1.0 MAY contain zero or more playlists. Additionally, the MPV playlist MAY reference assets in another MPV file. These features can be useful for applications that are using an
MPV file as a database. However, this usage is not recommended for MPV files that desire the highest interoperability because these features require advanced processing by the MPV reader.

For general interoperability, including use with consumer electronics devices, a MPV Music Profile file SHOULD have at most one playlist; in other words, only the first `<mpvp:Album>` element. Alternately, it is understood that only the first `<mpvp:Album>` element in a MPV Music Profile file MUST be processed by a conformant player.

### 4.1.2 Metadata Usage

Metadata placed on the `<mpvp:Album>` element has the scope of the whole album or playlist. For example, the `mpvm:PrincipalArtist` element can be used with either a single music asset or on the `<mpvp:Album>`.

<table>
<thead>
<tr>
<th>Album (&quot;Playlist&quot;) of Music</th>
<th><code>&lt;mpvp:Album&gt;</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Entries</td>
<td>implicit – number of entries in the foreground</td>
</tr>
<tr>
<td>Title</td>
<td>nmf:Metadata</td>
</tr>
<tr>
<td>Performed By</td>
<td>nmf:Metadata</td>
</tr>
<tr>
<td>Principal Artist</td>
<td>nmf:Metadata</td>
</tr>
<tr>
<td>Description</td>
<td>nmf:Metadata</td>
</tr>
<tr>
<td>Identifier</td>
<td>nmf:Metadata</td>
</tr>
<tr>
<td>Music Playlist Items</td>
<td>mpvp:Album</td>
</tr>
<tr>
<td>Playlist accompaniment</td>
<td>mpvp:Album</td>
</tr>
<tr>
<td>Key-Value Pairs</td>
<td>nmf:Metadata</td>
</tr>
<tr>
<td>Extra Data</td>
<td>mpv:Metadata and nmf:Metadata anywhere</td>
</tr>
</tbody>
</table>

**This content is played as the principal content of the music playlist.**

**This content is played as background to the music content in the foreground. For example, it could specify images.**

### 4.1.3 Background Usage

The `<mpvp:Background>` element is used in the MPV Music Profile to provide background presentation content to the music in the foreground. It is applied at the discretion of the playback application or device. If it is a simple device or application with no visual presentation capabilities, the `<mpvp:Background>` content SHOULD be ignored.

If the application or device has visual presentation capabilities, the `<mpvp:Background>` content may be processed. The recommended content is a series of `<mpv:StillRef>` elements that refer to still images that may be displayed behind the foreground content. This can be used in both browse/menu mode as well as playback mode. All the `[MPV-Pres]` properties may be applied, such as transitions. These are honored at the discretion of the playback application.

### 4.1.4 Foreground Usage

The MPV music playlist contains a foreground list (`<mpvp:Foreground>`) of music assets identified using the `<mpv:AudioRef>` element. This defines the set of primary music items in the playlist and their sequence. In addition to the referenced audio assets, the `<mpv:AssetList>` in the MPV file may also contain still image, video, and other kinds
of assets. The mpvp:Album makes it easy to identify which assets in the mpv:AssetList are primary by explicitly identifying them.

### 4.1.5 Related and Rendition Assets

A mpvp:Album may identify renditions and related assets. Album artwork may be specified using a mpv:Related element with an urn:osta-org:mpv:music:artwork relationship identifier. A pre-generated playback video of the whole album/playlist may be specified using a mpv:Rendition element with a mpv:renditionUsage="show".

### 4.2 Groups of Assets in Playlists

There may be situations in which a playlist has so many assets that it is inconvenient to navigate one item at a time. The playback application or device MAY choose to implement a FastForward or FastRewind type of behaviour that jumps ahead by multiples of items.

MPV-generating applications may also choose to group assets within playlists. The playback application MAY choose to implement FastForward or FastRewind behaviour by jumping ahead by groups; group navigation can also be explicit, if desired by the playback application or device.

Groups are specified using standard MPV functionality. A group of assets to be played sequentially is gathered together using the mpv:Seq asset. The playback behaviour of an application or device that encounters a mpv:Seq asset is to play the contents sequentially.

When mpv:Seqs are used with a mpvp:Album, only the mpv:Seq assets are listed in the album. Then processing application or device must then play the mpv:Seq contents.

**Example**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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:dc="http://purl.org/dc/elements/1.1/"
    xmlns:nmf="http://ns.osta.org/nmf/1.0/">
    <nmf:Metadata>
        <ManifestProperties xmlns="http://ns.osta.org/manifest/1.0/">
            <ProfileBag>
                <Profile>http://ns.osta.org/mpv/basic/1.0/</Profile>
                <Profile>http://ns.osta.org/mpv/presentation/1.0/</Profile>
                <Profile>http://ns.osta.org/mpv/music/1.0/</Profile>
            </ProfileBag>
        </ManifestProperties>
        <Properties xmlns="http://purl.org/dc/terms/">
            <created>2003-01-29T15:51:41Z</created>
            <modified>2003-01-29T15:51:41Z</modified>
        </Properties>
    </nmf:Metadata>
    <mpvp:Album>
        <nmf:Metadata>
            <dc:Properties>
                <dc:rights/>
                <dc:title>Piet's favorite Jazz</dc:title>
            </dc:Properties>
    </mpvp:Album>
```
<mpvm:MusicProperties>
  <mpvm:AlbumTitle/>  
  <mpvm:Genre>Jazz</mpvm:Genre>  
</mpvm:MusicProperties>

<nmf:Metadata>
  <Properties xmlns="http://purl.org/dc/elements/1.1">  
    <creator/>  
    <format>audio/mpeg</format>  
    <title>Ghostbuster's Theme.mp3</title>  
  </Properties>  
  <MusicProperties xmlns="http://ns.osta.org/mpv/music/1.0/">  
    <AlbumTitle>GhostBusters</AlbumTitle>  
    <Genre>theme</Genre>  
    <PlayingTime>246.48</PlayingTime>  
  </MusicProperties>  
</nmf:Metadata>

<mpv:LastURL>Ghostbuster's%20Theme.mp3</mpv:LastURL>

<mpv:Audio mpv:id="ID000002">
  <nmf:Metadata>
    <Properties xmlns="http://purl.org/dc/elements/1.1">  
      <creator/>  
      <format>audio/mpeg</format>  
      <title>Beach Boys</title>  
    </Properties>  
    <MusicProperties xmlns="http://ns.osta.org/mpv/music/1.0/">  
      <AlbumTitle>Made In USA</AlbumTitle>  
      <Genre/>  
      <PlayingTime>119.96</PlayingTime>  
    </MusicProperties>  
  </nmf:Metadata>
  <mpv:LastURL>409D.MP3</mpv:LastURL>
</mpv:Audio>

...
4.3 Linked Playlists

The [MPV-Core] specification established the structure and nomenclature of MPV files and assets. The MPV file is an XML document that is called an MPV Manifest and the outer-most element of a MPV file is a `<file:Manifest>`.

One very useful capability that [MPV-Core] provides is to link manifests to one another. The `<mpv:ManifestLink>` element creates a link to another MPV file. In this manner, just as with the WorldWideWeb, an endless chain of linked MPV files can be created.

Typically, when applied to a removable storage media like a CD, DVD, or memory card, all the links will be self-contained within the media. In this case, typically a file using the distinguished filename “index.mum” will contain a top-level list of linked playlists.

**EXAMPLE:**

In this example, a playlist consists only of links to three other playlists. Each playlist link has a screen-resolution image representation that can be used to enhance to a graphical presentation. Note that the filesystem location and names of these playlists is arbitrary and may be located at the choice of the authoring application or device.

Inspection of the linked playlists shows that they are specific orderings of the music – by genre, by artist, and by album. Thus the example is typical of a top-level playlist (e.g. an index.mum) which provides for access to other playlists with specific content organization. Each of the specific playlists could be further organized into additional playlists, e.g. one for each genre, artist, and album.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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>
    <ManifestProperties xmlns="http://ns.osta.org/manifest/1.0/">
      <InstanceID>23686AEFA3B340DAAC1BAF09B17DBCBC9</InstanceID>
      <ProfileBag>
        <Profile>http://ns.osta.org/mpv/basic/1.0/</Profile>
        <Profile>http://ns.osta.org/mpv/music/1.0/</Profile>
        <Profile>http://ns.osta.org/mpv/presentation/1.0/</Profile>
      </ProfileBag>
    </ManifestProperties>
    <mpvp:Album mpv:id="ALB001">
      <nmf:Metadata>
        <Properties xmlns="http://purl.org/dc/elements/1.1/">
          <creator>Pieter van Zee</creator>
          <description>A collection of my favorites songs from the 40’s and 50’s</description>
        </Properties>
        <Properties xmlns="http://purl.org/dc/terms/1.1/">
          <created>2002-12-01T23:11:00Z</created>
        </Properties>
      </nmf:Metadata>
      <mpvp:Background>
        <mpv:StillRef mpv:idRef="ID000000/"
      </mpvp:Background>
    </mpvp:Album>
  </nmf:Metadata>
</file:Manifest>
```
<mpv:ManifestLinkRef mpv:idRef="ID000100"/>
<mpv:ManifestLinkRef mpv:idRef="ID000200"/>
<mpv:ManifestLinkRef mpv:idRef="ID000300"/>
</mpvp:Foreground>
</mpvp:Album>

<mpv:AssetList>
<!-- background image for the playlist presentation -->
<mpv:Still mpv:id="ID000000">
<mpv:LastURL mpv:filesystem="Joliet">Playlists/Artwork/index.jpg</mpv:LastURL>
<mpv:LastURL mpv:filesystem="ISO9660-1">PLAYLIST/ARTWORK/INDEX.JPG</mpv:LastURL>
</mpv:Still>

<!-- ManifestLink -->
<mpv:ManifestLink mpv:id="ID000100">
<mpv:InstanceID>EF886AEFA3B340DA971BAF09B17DBC122</mpv:InstanceID>
<mpv:LastURL mpv:filesystem="Joliet">Playlists/by genre.mum</mpv:LastURL>
<mpv:LastURL mpv:filesystem="ISO9660-1">PLAYLIST/BY_GENRE.mum</mpv:LastURL>
<nmf:Metadata>
(Properties xmlns="http://purl.org/dc/elements/1.1/")
<title>By Genre</title>
</Properties>
</nmf:Metadata>
<mpv:Rendition mpv:renditionUsage="screen">
<mpv:StillRef mpv:idRef="ID000101"/>
</mpv:Rendition>
</mpv:ManifestLink>

<!-- thumbnail stand-in for the manifest -->
<mpv:Still mpv:id="ID000101">
<mpv:LastURL mpv:filesystem="Joliet">Playlists/Artwork/by genre.jpg</mpv:LastURL>
<mpv:LastURL mpv:filesystem="ISO9660-1">PLAYLIST/ARTWORK/BY_GENRE.JPG</mpv:LastURL>
</mpv:Still>

<!-- ManifestLink -->
<mpv:ManifestLink mpv:id="ID000200">
<mpv:InstanceID>34FE6AEFA3B340DA891BAF09B17DBC992</mpv:InstanceID>
<mpv:LastURL mpv:filesystem="Joliet">Playlists/by artist.mum</mpv:LastURL>
<mpv:LastURL mpv:filesystem="ISO9660-1">PLAYLIST/BY_ARTIS.mum</mpv:LastURL>
<nmf:Metadata>
(Properties xmlns="http://purl.org/dc/elements/1.1/")
<title>By Artist</title>
</Properties>
</nmf:Metadata>
<mpv:Rendition mpv:renditionUsage="screen">
<mpv:StillRef mpv:idRef="ID000201"/>
</mpv:Rendition>
</mpv:ManifestLink>

<!-- thumbnail stand-in for the manifest -->
<mpv:Still mpv:id="ID000201">
<mpv:LastURL mpv:filesystem="Joliet">Playlists/Artwork/by artist.jpg</mpv:LastURL>
<mpv:LastURL mpv:filesystem="ISO9660-1">PLAYLIST/ARTWORK/BY_ARTIS.JPG</mpv:LastURL>
</mpv:Still>

<!-- ManifestLink -->
<mpv:ManifestLink mpv:id="ID000300">
<mpv:InstanceID>93486AEFA3B340DA231BAF09B17DBCEFB</mpv:InstanceID>
<mpv:LastURL mpv:filesystem="Joliet">Playlists/by album.mum</mpv:LastURL>
<mpv:LastURL mpv:filesystem="ISO9660-1">PLAYLIST/BY_ALBUM.mum</mpv:LastURL>
</mpv:ManifestLink>

<!-- thumbnail stand-in for the manifest -->
<mpv:Still mpv:id="ID000301">
<mpv:LastURL mpv:filesystem="Joliet">Playlists/Artwork/by album.jpg</mpv:LastURL>
<mpv:LastURL mpv:filesystem="ISO9660-1">PLAYLIST/ARTWORK/BY_ALBUM.JPG</mpv:LastURL>
</mpv:Still>
This example would exist in the context of the following sample filesystem organization on a CD.

```
/index.mum
/Playlists/by album.mum
/Playlists/by artist.mum
/Playlists/by genre.mum
/Playlists/Artwork/by album.jpg
/Playlists/Artwork/by artist.jpg
/Playlists/Artwork/by genre.jpg
/Playlists/Artwork/index.jpg
```

### 4.4 Links to Foreign Playlists

Many music playlist formats exist. Since they are not in the MPV format, they are considered “foreign playlists”. MPV Music Profile allows links and references to foreign playlists, and applications and devices that understand foreign playlist formats can access them. In general, however, a MPV Music Profile player MUST understand only MPV playlists.

Foreign playlist assets are represented using the mpv:Document asset type. Note that mpv:ManifestLink assets must references files that are in the MPV format. When a foreground asset of the current playlist/album is a mpv:Document that is of a playlist type known to the MPV Music Profile reader, the reader SHOULD present the document as a link. If the link is selected by the user, the foreign playlist is loaded.

The type of playlist is recorded using nmf:Metadata|dc:Properties|format. The list of playlist media types is in Section 6.1, Music Manifest File Types & Extensions.

**EXAMPLE:**

In this example, a playlist consists only of links to two foreign M3U playlists. Note how the MPV wrapper is used to organize and represent information about the less-capable M3U playlists.
<mpv:Album mpv:id="ALB001">
  <nmf:Metadata>
    <Properties xmlns="http://purl.org/dc/elements/1.1/">
      <creator>Pieter van Zee</creator>
      <description>A collection of my favorites songs from the 40’s and 50’s</description>
      <title>Golden Oldies party music</title>
    </Properties>
    <Properties xmlns="http://purl.org/dc/terms/1.1/">
      <created>2002-12-01T23:11:00Z</created>
    </Properties>
  </nmf:Metadata>
  <mpv:Foreground>
    <mpv:DocumentRef mpv:idRef="ID000100"/>
    <mpv:DocumentRef mpv:idRef="ID000200"/>
  </mpv:Foreground>
</mpv:Album>

<mpv:AssetList>
  <!-- Document -->
  <mpv:Document mpv:id="ID000100">
    <mpv:InstanceID>EF886AEFA3B340DA971BAF09B17DBC122</mpv:InstanceID>
    <mpv:LastURL mpv:filesystem="Joliet">Playlists/by%20genre.m3u</mpv:LastURL>
    <mpv:LastURL mpv:filesystem="ISO9660-1">PLAYLIST/BY_GENRE.M3U</mpv:LastURL>
    <nmf:Metadata>
      <Properties xmlns="http://purl.org/dc/elements/1.1/">
        <format>audio/x-mpegurl</format>
        <title>By Genre</title>
      </Properties>
    </nmf:Metadata>
    <mpv:Rendition mpv:renditionUsage="screen">
      <mpv:StillRef mpv:idRef="ID000101"/>
    </mpv:Rendition>
  </mpv:ManifestLink>
  <!-- thumbnail stand-in for the manifest -->
  <mpv:Still mpv:id="ID000101">
    <mpv:LastURL mpv:filesystem="Joliet">Playlists/Artwork/by%20genre.jpg</mpv:LastURL>
    <mpv:LastURL mpv:filesystem="ISO9660-1">PLAYLIST/ARTWORK/BY_GENRE.JPG</mpv:LastURL>
  </mpv:Still>
  <!-- Document -->
  <mpv:Document mpv:id="ID000200">
    <mpv:InstanceID>34FE6AEFA3B340GAF09B17DBC992</mpv:InstanceID>
    <mpv:LastURL mpv:filesystem="Joliet">Playlists/by%20artist.m3u</mpv:LastURL>
    <mpv:LastURL mpv:filesystem="ISO9660-1">PLAYLIST/BY_ARTIS.M3U</mpv:LastURL>
    <nmf:Metadata>
      <Properties xmlns="http://purl.org/dc/elements/1.1/">
        <format>audio/x-mpegurl</format>
      </Properties>
    </nmf:Metadata>
  </mpv:ManifestLink>
  <!-- thumbnail stand-in for the manifest -->
  <mpv:Still mpv:id="ID000200">
    <mpv:LastURL mpv:filesystem="Joliet">Playlists/Artwork/by%20artist.jpg</mpv:LastURL>
    <mpv:LastURL mpv:filesystem="ISO9660-1">PLAYLIST/ARTWORK/BY_ARTIS.JPG</mpv:LastURL>
  </mpv:Still>
</mpv:AssetList>
This example would exist in the context of the following sample filesystem organization on a CD.

```
/index.mum
/Playlists/by artist.m3u
/Playlists/by genre.m3u
/Playlists/Artwork/by artist.jpg
/Playlists/Artwork/by genre.jpg
```

### 4.5 Default Playlists

When a MPV file has more than one playlist (mpv:Album), the default playlist is the first one in the file. For best interoperability, however, a MPV file has only one playlist. Instead, the default playlist behaviour is determined by the order in which playlists are processed by the application or device that is accessing them.

The default playlist is the first playlist that is encountered. Because the Music Profile specification fixed the algorithm by which MPV music manifest files are searched for, this is deterministic. *Index.mum* is the first file searched for, followed by *indexmum.xml*, *album.mum*, and *albummum.xml*. Then files with the .mum extension are processed in alphabetical order. If no file is found in the current directory, children, parent, and sibling directories are searched.

How the default playlist is processed is up to the processing application or device. The recommended best practices are that when the user is “browsing” or viewing a menu, the contents of the default list SHOULD be presented to the user. When the user is “playing/showing”, the contents of the list SHOULD be recursively played or interpreted. When a manifestLink is played, the behaviour is to open the file referenced by the link and play it.

For example, in “browse / menu” mode, the example file above would be presented to the user as a menu from which he could choose which playlist to use next. When in “playback” mode, the “By genre.mum” MPV file would be opened and it would be played.

### 4.6 Dynamic Playlists

So far, we have described how to create playlists of fixed collections of music. However, it is valuable to be able to represent the set of selection criteria that produced a given collection. This allows for round-trip editing and update of playlists and music collections when applied to an evolving collection of music over time.
Dynamic playlists also allow the more advanced player applications and devices to dynamically extend the playlists to reflect the music that is currently actually available. This is particularly valuable for hard-drive applications of MPV music playlists, where the user is regularly adding more music to a collection.

Imagine, for example, a harddrive-based music jukebox that contained your ever-growing complete collection of music. As you add new CDs to the collection, your jukebox playlists are automatically updated to include the new music selections. In addition, each time you insert a disc of music into the music jukebox, the MPV-aware jukebox offers to incrementally add more music to the disc from it’s collection that met the specified selection criteria of the playlists on the disc.

Support for dynamic playlists MAY be implemented in a maker or player application or device.

4.6.1 Specifying Dynamic Playlists

Dynamic playlists are specified essentially like database queries. They are stored as metadata on the mpvp:Foreground and mpvp:Background elements of an mpvp:Album. A processing application would apply them to the set of assets listed in the mpv:AssetList. Of course, this list may shrink or grow as more or less music is added to it.

Dynamic playlists can be specified with or without a set of static asset references. For the MPV Music Profile 1.0, a MPV playlist on removable storage media MUST include static asset references; it MAY include dynamic playlist queries that represent the selection criteria for the static assets that are specified.

4.6.2 Playlist Query Language

MPV Music Profile 1.0 allows applications to specify XPath queries that result in a list of assets that represent a MPV playlist. Since MPV is an XML-based grammar, it can use the widely adopted grammar called XPath [XPATH] that provides an industry-standard means to express selection criteria for one or more elements of an XML document.

Software SDKs that process XPath queries are widely available for most PC and server computing platforms and several multiplatform implementations are available. Low-end platforms, however, may not wish to support XPath queries because of the firmware and runtime memory requirements.

The XPath expression is executed relative to the <mpv:AssetList> node in the target MPV manifest file. The resulting list of assets that match the statement are treated for that instance as members of the playlist.

EXAMPLE

Todo

4.7 Best Practices for Generating Playlists

There are any number of ways for applications and devices to produce playlists. They may be the result of extensive manual user interaction. Or they may be generated automatically based on music metadata such as Principal Artist, Genre, and Album Title.

4.7.1 Top-Level Playlist

The most important best practice is to provide a top-level playlist, such as an index.mum. This gives the playback application or device the best ability to navigate contents in the order intended by the creator application.
4.7.2 “All Music” Playlist
It is RECOMMENDED that at least one playlist on a storage media play all the music. If this playlist is also the first playlist in the top-level playlist, it is the playlist that gets played first when a default playlist is used.

4.7.3 Local mpv:AssetList
Thus far, we have described putting the mpv:AssetList in each MPV playlist file. For broadest interoperability with all types of devices and applications, the assetlist MUST be in the same MPV files as the playlist (mpvp:Album).

4.7.4 Centralized mpv:AssetList
However, if interoperability is not the core objective, then an application or device MAY keep the playlists and assetlist in separate files. For example, a jukebox device such as the one described in Section 4.6, Dynamic Playlists, might store the mpv:AssetList representing the full set of music assets in a separate MPV file and create separate playlists expressed using dynamic playlists. This approach is efficient and scalable because it avoids duplicating information about the assets in more than one place. In addition to dynamic playlists, this approach can also accommodate static playlists by using the mpv:manifestLinkIdRef attribute in the mpv:AudioRef asset reference in the mpvp:Foreground or Background.

Usage of a centralized assetlist is defined by MPV Core and is quite practical for many applications, but it is not required of applications and devices that support the MPV Music Profile 1.0 specification for use on removable storage media because it requires a more capable playback device or application and additional computing resources. It is suitable for management of assets on non-removable media, such as a computer or device harddrive.
Chapter 5: MPV Music Schemas In Detail

5.1 Multiple Renditions of a Music Asset

The file specified in the primary asset should generally be the highest quality encoding while maintaining high compatibility with typical players. It may be useful to specify multiple renditions of a music asset. For example, if space is available on the storage media, multiple encodings of the same music asset can be specified, increasing the compatibility of the music disc with multiple players that may have different supported codecs and sustained throughputs.

Renditions of a music asset are specified using the <mpv:Rendition> tag and appropriate mpv:renditionUsage attribute values. For music asset renditions based on subsampling, the “subsampled” value of the mpv:renditionUsage attribute is the most appropriate. In addition, there may be multiple codec encodings of a given asset. These are encoding using the “alt” value of the mpv:renditionUsage attribute.

<table>
<thead>
<tr>
<th>RenditionUsage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alt</td>
<td>Alternate codec encoding of the same music asset. Metadata on the asset that is referenced can indicate the codec (nmf:Metadata</td>
</tr>
<tr>
<td>Subsampled</td>
<td>An encoding in the same codec as the primary asset but with a lesser bitrate. Metadata on the asset that is referenced can indicate the bitrate (nmf:Metadata</td>
</tr>
<tr>
<td>mpvm:EncodedBitrate</td>
<td>Describes the throughput in bits-per-second of the content as an integer value. For variable-bitrate-encoded assets, use the maximum value. For constant-bitrate-encoded assets, use the constant value. Typical values are 32000, 64000, 96000, 128000, 160000, 192000, 256000, 320000. Note that these values are not Kbps values, where K=1024, but kbps values where k=1000.</td>
</tr>
</tbody>
</table>
EXAMPLE

In this example, a primary MP3 file encoded at 160kbps has a MP3 file rendition encoded at 64kbps and a WMA file alternate rendition encoded at 64kbps.

```xml
<mpv:AssetList>
  <!-- This is the per-asset info -->
  <mpv:Audio mpv:id="ID01-GREAT-SWING-CLASSICS-20021202031833-a">
    <mpv:LastURL>01%20Great%20Swing%20Classics.mp3</mpv:LastURL>
    <nmf:Metadata>
      <dc:Properties>
        <dc:creator>Benny Goodman and his Orchestra</dc:creator>
        <dc:format>audio/mpeg</dc:format>
        <dc:title>Jumpin' At The Woodside</dc:title>
      </dc:Properties>
      <mpvm:MusicProperties>
        <mpvm:AlbumTitle>Great SWING CLASSICS in HI-FI</mpvm:AlbumTitle>
        <mpvm:Genre>Jazz</mpvm:Genre>
      </mpvm:MusicProperties>
    </nmf:Metadata>
  </mpv:Audio>

  <mpv:Rendition mpv:renditionUsage="subsampled">
    <mpv:AudioRef mpv:idRef="ID01-GREAT-SWING-CLASSICS-20021202031833-R1"/>
  </mpv:Related>

  <mpv:Related mpv:relationship="alt">
    <mpv:AudioRef mpv:idRef="ID01-GREAT-SWING-CLASSICS-20021202031833-R2"/>
  </mpv:Related>

  <mpv:Audio mpv:id="ID01-GREAT-SWING-CLASSICS-20021202031833-R1">
    <mpv:LastURL>01%20Great%20Swing%20Classics.mp3</mpv:LastURL>
    <nmf:Metadata>
      <dc:Properties>
        <dc:format>audio/mpeg</dc:format>
      </dc:Properties>
    </nmf:Metadata>
  </mpv:Audio>

  <mpv:Audio mpv:id="ID01-GREAT-SWING-CLASSICS-20021202031833-R2">
    <mpv:LastURL>01%20Great%20Swing%20Classics.wma</mpv:LastURL>
    <nmf:Metadata>
      <dc:Properties>
        <dc:format>audio/x-ms-wma</dc:format>
      </dc:Properties>
    </nmf:Metadata>
  </mpv:Audio>
</mpv:AssetList>
```
5.2 Artwork for an Asset

The MPV Music Profile allows rich types of artwork for an asset to be specified. A generating application can specify multiple kinds of artwork with any given music asset. The generating application has a choice: it can use the generic “relationship” string of “urn:osta-org:mpv:music:artwork”, or if available, a more specific relationship may be specified.

A processing application that doesn’t care about specific types of artwork can do a string match against the more generic “urn:osta-org:mpv:music:artwork” relationship, while more specific needs can also be matched. Note also that multiple related assets can be identified that refer to the same physical asset. For example, the cover artwork may also be artwork of a performance.

<table>
<thead>
<tr>
<th>Relationship type on an asset</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>urn:osta-org:mpv:music:artwork:coverFront</td>
<td>Front cover artwork of the media case containing this music asset</td>
</tr>
<tr>
<td>urn:osta-org:mpv:music:artwork:coverBack</td>
<td>Back cover artwork of the media case containing this music asset</td>
</tr>
<tr>
<td>urn:osta-org:mpv:music:artwork:leaflet</td>
<td>Front cover artwork of the media case containing this music asset</td>
</tr>
<tr>
<td>urn:osta-org:mpv:music:artwork:media</td>
<td>Artwork on the media containing this music asset</td>
</tr>
<tr>
<td>urn:osta-org:mpv:music:artwork:artist</td>
<td>Artwork depicting one or more artists of the music.</td>
</tr>
<tr>
<td>urn:osta-org:mpv:music:artwork:ensemble</td>
<td>Artwork depicting the performing ensemble, e.g. a band or orchestra.</td>
</tr>
<tr>
<td>urn:osta-org:mpv:music:artwork:conductor</td>
<td></td>
</tr>
<tr>
<td>urn:osta-org:mpv:music:artwork:performedBy</td>
<td></td>
</tr>
<tr>
<td>urn:osta-org:mpv:music:artwork:musicBy</td>
<td></td>
</tr>
<tr>
<td>urn:osta-org:mpv:music:artwork:lyricsBy</td>
<td></td>
</tr>
<tr>
<td>urn:osta-org:mpv:music:artwork:recordingLocation</td>
<td></td>
</tr>
<tr>
<td>urn:osta-org:mpv:music:artwork:recordingSession</td>
<td></td>
</tr>
<tr>
<td>urn:osta-org:mpv:music:artwork:performance</td>
<td></td>
</tr>
<tr>
<td>urn:osta-org:mpv:music:artwork:screenCapture</td>
<td></td>
</tr>
<tr>
<td>urn:osta-org:mpv:music:artwork:illustration</td>
<td></td>
</tr>
<tr>
<td>urn:osta-org:mpv:music:artwork:artistLogo</td>
<td></td>
</tr>
<tr>
<td>urn:osta-org:mpv:music:artwork:publisherLogo</td>
<td></td>
</tr>
<tr>
<td>urn:osta-org:mpv:music:artwork:thumbnail</td>
<td>32x32, GIF or PNG</td>
</tr>
</tbody>
</table>

**EXAMPLE**

In this example, a JPEG file is related to a MP3 file and identified as an image of a performance of the music asset.

```xml
...<mv:AssetList>
  <!-- This is the per-asset info -->
  <mv:Audio mpv:id="ID01-GREAT-SWING-CLASSICS-20021202031833-a">
    <mv:LastURL>01%20Great%20Swing%20Classics.mp3</mv:LastURL>
    <nmf:Metadata>
      <dc:Properties>
        <dc:creator>Benny Goodman and his Orchestra</dc:creator>
        <dc:format>audio/mpeg</dc:format>
        <dc:title>Jumpin' At The Woodside</dc:title>
      </dc:Properties>
    </nmf:Metadata>
  </mv:Audio>
</mv:AssetList>
```
5.3 Asset-related Content

The MPV Music Profile allows rich types of artwork for an asset to be specified. A generating application can specify multiple kinds of artwork with any given music asset. The generating application has a choice: it can use the generic “relationship” string of “urn:osta-org:mpv:music:artwork”, or if available, a more specific relationship may be specified.

| urn:osta-org:mpv:music:manuscript | The sheet music manuscript. Asset will be of Still, StillMultishotSequence, or Document. |

5.4 Music Identification

One of the three MPV core concepts is identification. Every asset can have zero or more mpv:ContentID elements which contain strongly-typed identifiers. [MPVCore] defines a basic identification scheme based on the MD5 algorithm to provide a statistically unique identifier. Other identification schemes may be defined by OSTA or any other organization.
OSTA MD5 Digital Signatures

The following informative discussion is derived from [MPVCore], which provides the normative reference. OSTA-defined digital signatures take the form

\[
\text{urn:osta-org:mpv:dsig:<algorithm>:<params>:<value>}
\]


Two MD5-hash-based digital signatures that are useful for music are:

\[
\text{urn:osta-org:mpv:dsig:md5:all:<value>}
\]

This signature includes every byte in the entire file.

\[
\text{urn:osta-org:mpv:dsig:md5:head:<byte count>:<value>}
\]

This signature includes only the first \(<\text{byte count}>\) integer number of bytes from the start of the file. It is attractive to robustly refer to very large files or files that are frequently edited or appended and for which the head can generate an approximately unique signature. If unspecified, the default byte count is 8192. Example: "urn:osta-org:mpv:dsig:md5:head:30000:EF886AEFA3B340da971BAF09B17DBC122"

\[
\text{urn:osta-org:mpv:dsig:md5:tail:<byte count>:<value>}
\]

This signature includes only the last \(<\text{byte count}>\) integer number of bytes from the end of the file. It is attractive to quickly detect changes in files that are frequently edited or appended. If unspecified, the default byte count is 8192. Example: "urn:osta-org:mpv:dsig:md5:tail:8192:EF886AEFA3B340da971BAF09B17DBC122"

In addition, a MD5 signature for the body of specific file types may be defined.

MusicCD Identifier Digital Signatures for Music

Specific audio tracks of a CDDA CD, generally known as a music CD, can be identified using an identifier described by [ID3v240] as the MusicCDIdentifier, MCDI. More information about the origins of and an algorithm for computing the MCDI is given in Appendix II: Music CD Identifier.

One digital signature format is defined by [ID3v240], known as a MusicCDIdentifier (MCDI). A sample implementation is provided by [FreeDB].

[ID3v240] defines the Music CD Identifier as follows:

This frame is intended for music that comes from a CD, so that the CD can be identified in databases such as the CDDB [CDDB]. The frame consists of a binary dump of the Table Of Contents, TOC, from the CD, which is a header of 4 bytes and then 8 bytes/track on the CD plus 8 bytes for the 'lead out' making a maximum of 804 bytes. The offset to the beginning of every track on the CD should be described with a four bytes absolute CD-frame address per track, and not with absolute time. This frame requires a present and valid "TRCK" frame, even if the CD's only got one track. There may only be one "MCDI" frame in each tag.

OSTA defines a ContentID that uses the MCDI, allowing a unique identifier for music published on a CD.

\[
\text{urn:osta-org:mpv:dsig:mcdi:<mcdi value in hex>:<track number>}
\]

This signature should allow music tracks that originate from a CD to be identified. Example: "urn:osta-org:mpv:dsig:mcdi:EF886AEFA3B340da971BAF09B17D:2"
OTHER DIGITAL SIGNATURES FOR MUSIC

A number of digital signatures for music have been defined. Organizations with specific algorithms may wish to define a digital signature syntax for use with MPV files. OSTA is able to maintain a list of defined digital signatures for general reference.

5.5 Music-specific Metadata

As identified by Section 3.5, Use of Dublin Core Metadata, some music metadata is recorded using the Dublin Core metadata properties. Additional music-specific properties are defined by the MPV Music Profile.

<table>
<thead>
<tr>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlbumTitle</td>
</tr>
<tr>
<td>ArrangedBy</td>
</tr>
<tr>
<td>EncodedBitrate</td>
</tr>
<tr>
<td>Genre</td>
</tr>
<tr>
<td>KeyValue</td>
</tr>
<tr>
<td>Lyrics</td>
</tr>
<tr>
<td>LyricsBy</td>
</tr>
<tr>
<td>Mood</td>
</tr>
<tr>
<td>MoreInfoURL</td>
</tr>
<tr>
<td>MusicBy</td>
</tr>
<tr>
<td>NumSets</td>
</tr>
<tr>
<td>NumTracks</td>
</tr>
<tr>
<td>PlayCount</td>
</tr>
<tr>
<td>PlayingTime</td>
</tr>
<tr>
<td>PrincipalArtist</td>
</tr>
<tr>
<td>ProducedBy</td>
</tr>
<tr>
<td>SetNumber</td>
</tr>
<tr>
<td>Situation</td>
</tr>
<tr>
<td>Tempo</td>
</tr>
<tr>
<td>TrackNumber</td>
</tr>
</tbody>
</table>

They are specified as NMF metadata, meaning that they are conformant to the [NMF] specification. NMF requires all metadata elements to be organized alphabetically and has other constraints.

General Usage is as subelements of the <mpvm:MusicProperties> element.

```xml
<mpv:Audio mpv:id="ID01-GREAT-SWING-CLASSICS-20021202031833-a">
  ...
  <nmf:Metadata>
    ...
    <mpvm:MusicProperties>
      ...
      </mpvm:MusicProperties>
    </nmf:Metadata>
  ...
</mpv:Audio>
```

MPVM:MusicProperties
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://ns.osta.org/mpv/music/1.0/
xmlns="http://ns.osta.org/mpv/music/1.0/" xmlns:nmft="http://ns.osta.org/nmf/1.0/tools/
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:nmf="http://ns.osta.org/nmf/1.0/
xmlns:mpvmLyric="http://ns.osta.org/mpv/music/1.0/lyric/" elementFormDefault="qualified"
attributeFormDefault="qualified">
  <xs:import namespace="http://ns.osta.org/nmf/1.0/" schemaLocation="../../imports/nmf/base.xsd"/>
  <xs:annotation>
    <xs:documentation>The MPV Music Properties schema</xs:documentation>
  </xs:annotation>
  <!--
  name for BySchemaProperties element
  -->
  <xs:element name="MusicProperties" type="BySchemaPropsType"
substitutionGroup="nmf:BySchemaPropsBase"/>
  <!--
  top-level schema element type
  -->
  <xs:complexType name="BySchemaPropsType">
    <xs:complexContent>
      <xs:extension base="nmf:BySchemaPropsBase">
        <xs:sequence>
          <xs:element ref="AlbumTitle" minOccurs="0"/>
          <xs:element ref="ArrangedBy" minOccurs="0"/>
          <xs:element ref="EncodedBitrate" minOccurs="0"/>
          <xs:element ref="Genre" minOccurs="0"/>
          <xs:element ref="KeyValue" minOccurs="0"/>
          <xs:element ref="Lyrics" minOccurs="0"/>
          <xs:element ref="LyricsBy" minOccurs="0"/>
          <xs:element ref="Mood" minOccurs="0"/>
          <xs:element ref="MoreInfoURL" minOccurs="0"/>
          <xs:element ref="MusicBy" minOccurs="0"/>
          <xs:element ref="NumSets" minOccurs="0"/>
          <xs:element ref="NumTracks" minOccurs="0"/>
          <xs:element ref="PlayCount" minOccurs="0"/>
          <xs:element ref="PlayingTime" minOccurs="0"/>
          <xs:element ref="PrincipalArtist" minOccurs="0"/>
          <xs:element ref="ProducedBy" minOccurs="0"/>
          <xs:element ref="Recorded" minOccurs="0"/>
          <xs:element ref="SetNumber" minOccurs="0"/>
          <xs:element ref="Situation" minOccurs="0"/>
          <xs:element ref="Tempo" minOccurs="0"/>
          <xs:element ref="TrackNumber" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="AlbumTitle" type="AlbumTitleType"/>
  <xs:complexType name="AlbumTitleType">
    <xs:simpleContent>
      <xs:extension base="xs:string"/>
    </xs:simpleContent>
  </xs:complexType>
</xs:schema>
<xs:element name="ArrangedBy" type="ArrangedByType"/>
<xs:complexType name="ArrangedByType">
    <xs:simpleContent>
        <xs:extension base="xs:string"/>
    </xs:simpleContent>
</xs:complexType>

<xs:element name="EncodedBitrate" type="EncodedBitrateType"/>
<xs:complexType name="EncodedBitrateType">
    <xs:simpleContent>
        <xs:extension base="xs:int"/>
    </xs:simpleContent>
</xs:complexType>

<xs:element name="Genre" type="GenreType"/>
<xs:simpleType name="GenreType">
    <xs:union memberTypes="GenreBaseType xs:anyURI"/>
</xs:simpleType>
<xs:simpleType name="GenreBaseType">
    <xs:restriction base="xs:string">
    <!-- Consistent with ID3v1 Genre from id3.org -->
    <xs:enumeration value="Blues"/>  <!-- ID3v1 Genre # 0. -->
    <xs:enumeration value="ClassicRock"/>  <!-- ID3v1 Genre # 1. -->
    <xs:enumeration value="Country"/>  <!-- ID3v1 Genre # 2. -->
    <xs:enumeration value="Dance"/>  <!-- ID3v1 Genre # 3. -->
    <xs:enumeration value="Disco"/>  <!-- ID3v1 Genre # 4. -->
    <xs:enumeration value="Funk"/>  <!-- ID3v1 Genre # 5. -->
    <xs:enumeration value="Grunge"/>  <!-- ID3v1 Genre # 6. -->
    <xs:enumeration value="Hip-Hop"/>  <!-- ID3v1 Genre # 7. -->
    <xs:enumeration value="Jazz"/>  <!-- ID3v1 Genre # 8. -->
    <xs:enumeration value="Metal"/>  <!-- ID3v1 Genre # 9. -->
    <xs:enumeration value="New Age"/>  <!-- ID3v1 Genre # 10. -->
    <xs:enumeration value="Oldies"/>  <!-- ID3v1 Genre # 11. -->
    <xs:enumeration value="Other"/>  <!-- ID3v1 Genre # 12. -->
    <xs:enumeration value="Pop"/>  <!-- ID3v1 Genre # 13. -->
    <xs:enumeration value="R&B"/>  <!-- ID3v1 Genre # 14. -->
    <xs:enumeration value="Rap"/>  <!-- ID3v1 Genre # 15. -->
    <xs:enumeration value="Reggae"/>  <!-- ID3v1 Genre # 16. -->
    <xs:enumeration value="Rock"/>  <!-- ID3v1 Genre # 17. -->
    <xs:enumeration value="Techno"/>  <!-- ID3v1 Genre # 18. -->
    <xs:enumeration value="Industrial"/>  <!-- ID3v1 Genre # 19. -->
    <xs:enumeration value="Alternative"/>  <!-- ID3v1 Genre # 20. -->
    <xs:enumeration value="SkA"/>  <!-- ID3v1 Genre # 21. -->
    <xs:enumeration value="Death Metal"/>  <!-- ID3v1 Genre # 22. -->
    <xs:enumeration value="Pranks"/>  <!-- ID3v1 Genre # 23. -->
    <xs:enumeration value="Soundtrack"/>  <!-- ID3v1 Genre # 24. -->
    <xs:enumeration value="Euro-Techno"/>  <!-- ID3v1 Genre # 25. -->
    <xs:enumeration value="Ambient"/>  <!-- ID3v1 Genre # 26. -->
    <xs:enumeration value="Trip-Hop"/>  <!-- ID3v1 Genre # 27. -->
    <xs:enumeration value="Vocal"/>  <!-- ID3v1 Genre # 28. -->
    <xs:enumeration value="Jazz+Funk"/>  <!-- ID3v1 Genre # 29. -->
    <xs:enumeration value="Fusion"/>  <!-- ID3v1 Genre # 30. -->
    <xs:enumeration value="Trance"/>  <!-- ID3v1 Genre # 31. -->
    <xs:enumeration value="Classical"/>  <!-- ID3v1 Genre # 32. -->
    <xs:enumeration value="Instrumental"/>  <!-- ID3v1 Genre # 33. -->
    <xs:enumeration value="Acid"/>  <!-- ID3v1 Genre # 34. -->

Ver. 0.90WD, 21 July 2003 © 2002-2003 OSTA 41 of 77
<xs:enumeration value="House"/>
<xs:enumeration value="Game"/>
<xs:enumeration value="Sound Clip"/>
<xs:enumeration value="Gospel"/>
<xs:enumeration value="Noise"/>
<xs:enumeration value="AlternRock"/>
<xs:enumeration value="Bass"/>
<xs:enumeration value="Soul"/>
<xs:enumeration value="Punk"/>
<xs:enumeration value="Space"/>
<xs:enumeration value="Meditative"/>
<xs:enumeration value="Instrumental Pop"/>
<xs:enumeration value="Instrumental Rock"/>
<xs:enumeration value="Ethnic"/>
<xs:enumeration value="Gothic"/>
<xs:enumeration value="Darkwave"/>
<xs:enumeration value="Techno-Industrial"/>
<xs:enumeration value="Electronic"/>
<xs:enumeration value="Pop-Folk"/>
<xs:enumeration value="Eurodance"/>
<xs:enumeration value="Dream"/>
<xs:enumeration value="Southern Rock"/>
<xs:enumeration value="Comedy"/>
<xs:enumeration value="Cult"/>
<xs:enumeration value="Gangsta"/>
<xs:enumeration value="Top 40"/>
<xs:enumeration value="Christian Rap"/>
<xs:enumeration value="Pop/Funk"/>
<xs:enumeration value="Jungle"/>
<xs:enumeration value="Native American"/>
<xs:enumeration value="Cabaret"/>
<xs:enumeration value="New Wave"/>
<xs:enumeration value="Psychadelic"/>
<xs:enumeration value="Rave"/>
<xs:enumeration value="Showtunes"/>
<xs:enumeration value="Trailer"/>
<xs:enumeration value="Lo-Fi"/>
<xs:enumeration value="Tribal"/>
<xs:enumeration value="Acid Punk"/>
<xs:enumeration value="Acid Jazz"/>
<xs:enumeration value="Polka"/>
<xs:enumeration value="Retro"/>
<xs:enumeration value="Musical"/>
<xs:enumeration value="Rock & Roll"/>
<xs:enumeration value="Hard Rock"/>

<!-- Consistent with ID3v1 Genre Extensions -->
<xs:enumeration value="Folk"/>
<xs:enumeration value="Folk-Rock"/>
<xs:enumeration value="National Folk"/>
<xs:enumeration value="Swing"/>
<xs:enumeration value="Fast Fusion"/>
<xs:enumeration value="Bebob"/>
<xs:enumeration value="Latin"/>
<xs:enumeration value="Revival"/>
<xs:enumeration value="Celtic"/>
<xs:enumeration value="Bluegrass"/> <!-- ID3v1 Genre Extension # 89. -->
<xs:enumeration value="Avantgarde"/>  <!-- ID3v1 Genre Extension # 90. -->
<xs:enumeration value="Gothic Rock"/>  <!-- ID3v1 Genre Extension # 91. -->
<xs:enumeration value="Progressive Rock"/>  <!-- ID3v1 Genre Extension # 92. -->
<xs:enumeration value="Psychedelic Rock"/>  <!-- ID3v1 Genre Extension # 93. -->
<xs:enumeration value="Symphonic Rock"/>  <!-- ID3v1 Genre Extension # 94. -->
<xs:enumeration value="Slow Rock"/>  <!-- ID3v1 Genre Extension # 95. -->
<xs:enumeration value="Big Band"/>   <!-- ID3v1 Genre Extension # 96. -->
<xs:enumeration value="Chorus"/>  <!-- ID3v1 Genre Extension # 97. -->
<xs:enumeration value="Easy Listening"/>  <!-- ID3v1 Genre Extension # 98. -->
<xs:enumeration value="Acoustic"/>  <!-- ID3v1 Genre Extension # 99. -->
<xs:enumeration value="Humour"/>   <!-- ID3v1 Genre Extension #100. -->
<xs:enumeration value="Speech"/>   <!-- ID3v1 Genre Extension #101. -->
<xs:enumeration value="Chanson"/>  <!-- ID3v1 Genre Extension #102. -->
<xs:enumeration value="Opera"/>   <!-- ID3v1 Genre Extension #103. -->
<xs:enumeration value="Chamber Music"/>  <!-- ID3v1 Genre Extension #104. -->
<xs:enumeration value="Sonata"/>  <!-- ID3v1 Genre Extension #105. -->
<xs:enumeration value="Symphony"/>  <!-- ID3v1 Genre Extension #106. -->
<xs:enumeration value="Booty Bass"/>  <!-- ID3v1 Genre Extension #107. -->
<xs:enumeration value="Primus"/>  <!-- ID3v1 Genre Extension #108. -->
<xs:enumeration value="Porn Groove"/>  <!-- ID3v1 Genre Extension #109. -->
<xs:enumeration value="Satire"/>   <!-- ID3v1 Genre Extension #110. -->
<xs:enumeration value="Slow Jam"/>  <!-- ID3v1 Genre Extension #111. -->
<xs:enumeration value="Club"/>    <!-- ID3v1 Genre Extension #112. -->
<xs:enumeration value="Tango"/>   <!-- ID3v1 Genre Extension #113. -->
<xs:enumeration value="Samba"/>   <!-- ID3v1 Genre Extension #114. -->
<xs:enumeration value="Folklore"/>  <!-- ID3v1 Genre Extension #115. -->
<xs:enumeration value="Ballad"/>   <!-- ID3v1 Genre Extension #116. -->
<xs:enumeration value="Power Ballad"/>  <!-- ID3v1 Genre Extension #117. -->
<xs:enumeration value="Rhythmic Soul"/>  <!-- ID3v1 Genre Extension #118. -->
<xs:enumeration value="Freestyle"/>  <!-- ID3v1 Genre Extension #119. -->
<xs:enumeration value="Duet"/>    <!-- ID3v1 Genre Extension #120. -->
<xs:enumeration value="Punk Rock"/>  <!-- ID3v1 Genre Extension #121. -->
<xs:enumeration value="Drum Solo"/>  <!-- ID3v1 Genre Extension #122. -->
<xs:enumeration value="A capella"/>  <!-- ID3v1 Genre Extension #123. -->
<xs:enumeration value="Euro-House"/>  <!-- ID3v1 Genre Extension #124. -->
<xs:enumeration value="Dance-House"/>  <!-- ID3v1 Genre Extension #125. -->

</xs:restriction>
</xs:simpleType>

<xs:element name="LyricsBy" type="LyricsByType"/>
<xs:complexType name="LyricsByType">
  <xs:simpleContent>
    <xs:extension base="xs:string"/>
  </xs:simpleContent>
</xs:complexType>

<xs:element name="Lyrics" type="LyricsType"/>
<xs:complexType name="LyricsType">
  <xs:complexContent>
    <xs:element name="Lang" type="xs:string" minOccurs="0"/>
    <xs:element ref="LyricPart" type="LyricPartType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:complexContent>
</xs:complexType>
<xs:element name="LyricPart" type="LyricPartType"/>
<xs:complexType name="LyricPartType">
  <xs:complexContent>
    <xs:element name="TimeOffset" type="xs:float" minOccurs="0" />
    <xs:element name="Text" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
  </xs:complexContent>
</xs:complexType>

<xs:element name="MoreInfoURL" type="MoreInfoURLType"/>
<xs:complexType name="MoreInfoURLType">
  <xs:simpleContent>
    <xs:extension base="xs:string"/>
  </xs:simpleContent>
</xs:complexType>

<xs:element name="Mood" type="MoodType"/>
<xs:complexType name="MoodType">
  <xs:simpleContent>
    <xs:extension base="xs:string"/>
  </xs:simpleContent>
</xs:complexType>

<xs:element name="MusicBy" type="MusicByType"/>
<xs:complexType name="MusicByType">
  <xs:simpleContent>
    <xs:extension base="xs:string"/>
  </xs:simpleContent>
</xs:complexType>

<xs:element name="NumSets" type="NumSetsType"/>
<xs:complexType name="NumSetsType">
  <xs:simpleContent>
    <xs:extension base="xs:int"/>
  </xs:simpleContent>
</xs:complexType>

<xs:element name="NumTracks" type="NumTracksType"/>
<xs:complexType name="NumTracksType">
  <xs:simpleContent>
    <xs:extension base="xs:int"/>
  </xs:simpleContent>
</xs:complexType>

<xs:element name="PlayCount" type="PlayCountType"/>
<xs:complexType name="PlayCountType">
  <xs:simpleContent>
    <xs:extension base="xs:int"/>
  </xs:simpleContent>
</xs:complexType>

<xs:element name="PrincipalArtist" type="PrincipalArtistType"/>
<xs:complexType name="PrincipalArtistType">
  <xs:simpleContent>
    <xs:extension base="xs:string"/>
  </xs:simpleContent>
</xs:complexType>
5.6 Album/Playlist-level <mpvm:MusicProperties> Music Metadata

The Music Profile defines a schema for music properties. This schema can be used on all audio assets by specifying the root element of the mpvm schema as the only child of the nmf:Metadata element.

The guiding practice for applications and devices that process and present MPV music content based on this schema is that music properties on an mpvp:Album apply also to the tracks contained by that album.

5.7 <mpvm:AudioWithStills> Music and Stills Asset

The MPV Music Profile defines a new composite asset type, AudioWithStills. The AudioWithStills asset is the counterpart to the StillWithAudio asset defined by [MPVCore]. AudioWithStills can be used to define a composite playback experience in which an Audio asset is played “in the foreground”, while Still assets are displayed according to the presentation specified.

One use of this asset type is for synchronized music and visual display, such as for karaoke or presentations in which the audio display is primary.

Example:

```xml
...<mpv:AssetList>
  <mpv:AudioWithStills mpv:id="ID000100">
    <mpv:InstanceID>AB893AF0A33B40AD971BFA09B17DBC193</mpv:InstanceID>
    <nmf:Metadata>
      <Properties xmlns="http://purl.org/dc/elements/1.1/">
        <format>image/tiff</format>
        <title>June 9, 2002, 14:34</title>
      </Properties>
    </nmf:Metadata>
    <mpv:AudioRef mpv:idRef="ID000101"/>
    <mpv:StillRef mpv:idRef="ID000102"/>
    <mpv:StillRef mpv:idRef="ID000103"/>
  </mpv:AudioWithStills>
  <mpv:Audio mpv:id="ID001200">
    <mpv:ContentID>urn:osta-grademark:ID001200</mpv:ContentID>
    <mpv:LastURL>Songs/My Gal.mp3</mpv:LastURL>
  </mpv:Audio>
  <mpv:Still mpv:id="ID000100">
    <mpv:ContentID>urn:osta-grademark:ID000100</mpv:ContentID>
    <mpv:LastURL>screem/Slide1.JPG</mpv:LastURL>
  </mpv:Still>
  <mpv:Still mpv:id="ID000101">
    <mpv:ContentID>urn:osta-grademark:ID000101</mpv:ContentID>
    <mpv:LastURL>screem/Slide2.JPG</mpv:LastURL>
  </mpv:Still>
</mpv:AssetList>...
```
element `mpv:AudioWithStills`, complexType `mpv:AudioWithStillsType`

namespace `http://ns.osta.org/mpv/music/1.0/`

type `mpv:AudioWithStillsType`


attributes

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Use</th>
<th>Default</th>
<th>Fixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>mpv:id</td>
<td>xs:ID</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

source `<xs:element name="AudioWithStills" type="mpv:AudioWithStillsType" substitutionGroup="mpv:CompositeAssetBase"/>

source `<xs:complexType name="AudioWithStillsType">
  `<xs:complexContent>
    `<xs:extension base="mpv:CompositeAssetBaseType">
      `<xs:sequence>
        `<xs:element ref="mpv:AudioRef"/>
        `<xs:element ref="mpv:StillRef" minOccurs="unbounded"/>
        `<xs:element name="DisplayRate" type="xs:string" minOccurs="0"/>
      `<xs:group ref="mpv:RelationsElemGroup"/>
    `<xs:sequence>
  `<xs:extension>
  `<xs:complexContent>
`  `<xs:complexType>

**DisplayRate**

The value of DisplayRate is a sequence of still-to-still durations that indicate the display rate. The semicolon character “;” is used as a delimiter and the path begins with an algorithm declaration. The only rate algorithm defined by MPV is "FrameToFrame".

The frame to frame algorithm uses the following DisplayRate syntax described in BNF. This BNF conforms to the BNF usage from IETF specifications such as in [URI]. Clock value is always in relative time in seconds to the previous frame.

\[
\text{DisplayRate} = \text{"FrameToFrame:" (\text{offset-clock-value} ;\text{frame-clock-value})? <frame-clock-value>}
\]

\[
\text{offset-clock-value} = \text{"o" <clock-value>}
\]

\[
\text{frame-clock-value} = <\text{clock-value} ( ;\text{clock-value})^*\text{clock-value}>\]

\[
\text{clock-value} = <\text{decimal number} (>\text{decimal number})^*\text{decimal number}>\]

\[
\text{decimal number} = [0-9] [0-9]^*
\]

There are as many as N clock values for N images. N clock values are possible and not N-1 because the first value optionally can be used to indicate the offset time between an arbitrary timepoint and when the first frame is captured. The last value provided is reused for all subsequent durations.

Example:

"FrameToFrame:0.3": any number of still images, each 0.3 seconds after the previous.

"FrameToFrame:0;0.4;0.4": 4 images, each 0.4 seconds after the previous.

"FrameToFrame:0.3;0.4;0.4": 4 images, the first 3.1 seconds from when timing began, each frame 0.4 seconds after the previous.

"FrameToFrame:120;210;70": 4 images, the second taken 120 seconds after the first, the third taken 210 seconds after the second, the fourth taken 70 seconds after the third.
5.8 MPV Music Profile Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<file:Manifest xmlns:file="http://ns.osta.org/manifest/1.0/"
    xmlns:mpv="http://ns.osta.org/mpv/1.0/"
    xmlns:mpvp="http://ns.ota.org/mpv/presentation/1.0/"
    xmlns:mpvm="http://ns.osta.org/mpv/music/1.0/"
    xmlns:dc="http://ns.ota.org/nmf/1.0/dc/"
    xmlns:nmf="http://ns.ota.org/nmf/1.0/"
    xmlns:nmff="http://ns.ota.org/nmf/1.0/"
    xmlns:manifestProperties="http://ns.ota.org/manifest/1.0/"/>
```
http://ns.osta.org/mpv/basic/1.0/
http://ns.osta.org/mpv/presentation/1.0/
http://ns.osta.org/mpv/music/1.0/

<ProfileBag>
<Profile>http://ns.osta.org/mpv/basic/1.0/</Profile>
<Profile>http://ns.osta.org/mpv/presentation/1.0/</Profile>
<Profile>http://ns.osta.org/mpv/music/1.0/</Profile>
</ProfileBag>
</ManifestProperties>
</nmf:Metadata>
<mpvp:Album>
<nmf:Metadata>
<dc:Properties>
<dc:description>14 swing classics re-recorded in the '50s by the original artists for great sound with all the integrity and excitement of the original performances.</dc:description>
<dc:identifier>7243 5 21223 2 5 Capitol Jazz</dc:identifier>
<dc:rights>(P) and (C) 1999 Capitol Records, Inc. All rights reserved.</dc:rights>
<dc:title/>
</dc:Properties>
<mpvm:MusicProperties>
<mpvm:AlbumTitle>Great SWING CLASSICS in HI-FI</mpvm:AlbumTitle>
<mpvm:Genre>Jazz</mpvm:Genre>
</mpvm:MusicProperties>
</nmf:Metadata>
<mpvp:Foreground>
<mpv:AudioRef mpv:idRef="01-GREAT-SWING-CLASSICS-20021202031833-a"/>
<mpv:AudioRef mpv:idRef="02-GREAT-SWING-CLASSICS-20021202031833-a"/>
</mpvp:Foreground>
</mpvp:Album>
<mpv:AssetList>
<mpv:Audio mpv:id="01-GREAT-SWING-CLASSICS-20021202031833-a">
<mpv:LastURL>01%20Great%20Swing%20Classics.wma</mpv:LastURL>
<nmf:Metadata>
<dc:Properties>
<dc:creator>Benny Goodman and his Orchestra</dc:creator>
<dc:description/>
<dc:format>audio/x-ms-wma</dc:format>
<dc:identifier/>
<dc:title>Jumpin' At The Woodside</dc:title>
</dc:Properties>
<mpvm:MusicProperties>
<mpvm:AlbumTitle>Great SWING CLASSICS in HI-FI</mpvm:AlbumTitle>
<mpvm:Genre>Jazz</mpvm:Genre>
<mpvm:MusicBy>Count Basie</mpvm:MusicBy>
<mpvm:ArrangedBy>Count Basie;Jimmy Mundy</mpvm:ArrangedBy>
<mpvm:NumTracks>14</mpvm:NumTracks>
<mpvm:PlayingTime>208.12</mpvm:PlayingTime>
<mpvm:PrincipalArtist>Benny Goodman</mpvm:PrincipalArtist>
<mpvm:Recorded>1954-11-09</mpvm:Recorded>
<mpvm:TrackNumber>1</mpvm:TrackNumber>
</mpvm:MusicProperties>
</nmf:Metadata>
</mpv:Audio>
<mpv:Audio mpv:id="02-GREAT-SWING-CLASSICS-20021202031833-a">
<mpv:LastURL>02%20Great%20Swing%20Classics.wma</mpv:LastURL>
<nmf:Metadata>
<dc:Properties>
<dc:creator>Duke Ellington and his Orchestra</dc:creator>
</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:ArrangedBy>Duke Ellington</mpvm:ArrangedBy>
<mpvm:NumTracks>14</mpvm:NumTracks>
<mpvm:PlayingTime>208.12</mpvm:PlayingTime>
<mpvm:PrincipalArtist>Duke Ellington</mpvm:PrincipalArtist>
<mpvm:Recorded>1931-02-10</mpvm:Recorded>
<mpvm:TrackNumber>1</mpvm:TrackNumber>
</mpvm:MusicProperties>
</nmf:Metadata>
</mpv:Audio>
<dc:description/>
<dc:format>audio/x-ms-wma</dc:format>
<dc:identifier/>
<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>236.36</mpvm:PlayingTime>
  <mpvm:PrincipalArtist>Duke Ellington</mpvm:PrincipalArtist>
  <mpvm:Recorded>1955-11-17</mpvm:Recorded>
  <mpvm:TrackNumber>2</mpvm:TrackNumber>
</mpvm:MusicProperties>
</nmf:Metadata>
</mpv:Audio>
</mpv:AssetList>
</file:Manifest>
Chapter 6: MPV Music Profile Extensions to MPV Core Specification

6.1 Music Manifest File Types & Extensions

For systems in which file type is carried by the file name extension, such as Microsoft Windows and Unix, the Music Manifest file will utilize an extension. The MPV Music Profile defines two extensions a manifest may carry.

.mum

This extension identifies a file to be a MUsic Manifest (MUM).

Usage is case insensitive. This extension may be registered by an application to provide default and alternate processors of Music Manifests.

.xml

This extension identifies a file as containing XML content. Usage is case insensitive. A Music Manifest should only use this extension if it expects to be processed by a general-purpose XML processor such as Microsoft Internet Explorer. It is recommended that the manifest include an XML processing instruction specifying a stylesheet to use for presentation.

This extension may be registered by an application to provide general purpose XML content processing. An application should register this extension with care, as many types of content may carry the .xml extension and an application should do its best to handle this content in a general fashion.

For example, Microsoft Internet Explorer 5.5 and above registers this extension; when it processes the file, it looks for a stylesheet processing instruction. IE renders the results of applying the stylesheet to the XML content. This separation of content and presentation allows IE to be a general purpose XML processing engine and suitable for handling the .xml extension.

The Apple Macintosh operating system uses an internal file type stored as a resource value of the data fork of a file. The following file type may be used for Music Manifests on Macintosh systems. Apple no longer requires file type registration.

.mum

This Apple Macintosh file type identifies the file to contain a Music Manifest. Usage is case sensitive. This extension may be registered by an application to provide a default processor of Music Manifests.

Some applications examine leading characters of a file in an attempt to determine its file type. No byte sequences can be counted on to always be present, but generally all XML documents in the UTF-8 charsets begin with hexadecimal 3C 3F 78 6D 6C, ("<?xml"). While this will identify the document as an XML document, it does NOT
identify it as an Music Manifest. This requires parsing the document to locate the outer element defined by the manifest schema.

### 6.2 Music Manifest MIME Media Type

MIME media types are widely used in internet applications to indicate the type of a file or content in a manner external of the file and independent of the name of the file or any information embedded in the file [MIME-2]. IANA maintains a registry of MIME media types and the set of MIME media types IANA thinks is registered at any time can be found at [MIMETYPES-REG].

The MIME media types that can be used for a Music Manifest are:

**application/vnd.osta-org.mum+xml**

This MIME media type identifies the content to be a Music Manifest. Usage is case sensitive. This media type may be registered with internet browsers by an application to provide the default processor of a Music Manifest.

**application/xml**

This MIME media type identifies the content as containing XML content. Usage is case sensitive. A Music Manifest should only use this MIME type if it expects to be processed by a general-purpose XML processor such as Microsoft Internet Explorer. It is recommended that the manifest include an XML processing instruction specifying a stylesheet to use for presentation.

This MIME media type may be registered by an application to provide general purpose XML content processing. An application should register this media type with care, as many types of content may carry the application/xml media type and an application should do its best to handle this content in a general fashion.

For example, Microsoft Internet Explorer 5.5 and above registers this media type; when it processes the file, it looks for a stylesheet processing instruction. IE renders the results of applying the stylesheet to the XML content. This separation of content and presentation allows IE to be a general purpose XML processing engine and suitable for handling the .xml extension.

### 6.3 Choosing Which File Type and MIME Media Type to Use

For products authoring Music Manifests, the choice of file extension and MIME media type is important. The product should consider the contexts in which it expects the manifest to be used. The primary decision factor is whether the product expects the manifest to be used in an environment that is explicitly MPV-aware or one that is not.

A MPV-aware environment will have the .mum file extension and application/vnd.osta-org.mum+xml media type registered to an application. A MPV-unaware environment will not.

Generally speaking, it is preferable to use a Music Manifest in an MPV-aware environment because the MPV-aware application is better able to utilize fully the MPV capabilities. In particular, an MPV-aware environment will likely handle better the situation in which the default lastURL reference in invalid; it should use other available lastURL values or the identifiers available on an asset to fixup the lastURL value.
### 6.4 Finding a Music Manifest File

The Music Manifest is the essential document to be managed and manipulated for collections of music content. MPV collections define a structured association of assets and provide access to metadata about those assets.

When searching a file system for a Music Manifest, they can be located by name or by extension. When requested by name, the manifest is either found or not found. If not found, the algorithm defined elsewhere for lastURL fixup should be applied.

The MPV Music Profile defines the following algorithm that describes how to locate a MPV manifest when no name of one is known.

If dealing with a removable storage unit, e.g. an optical disc inserted, the starting current working directory is the root directory.

If dealing with a user's personal computer "login" account, there may be a set of directories to be considered in sequence that will lead to the "root" Music Manifest for the account. Best Practices for which directories to consider are defined elsewhere.

If browsing a filesystem, the current working directory is decided by the application conducting the search.

The scan algorithm to find a Music Manifest from a given current working directory is:

```
In the current working directory, look for a file with one of the following case-insensitive names according to the order given.
INDEX.MUM
INDEXMUM.XML
ALBUM.MUM
ALBUMMUM.XML
<any name>.MUM, in an undefined order when more than one is present

If no matching file is found, the child directories of the current directory are scanned in an alphabetical breadth-first traversal to a depth of one subdirectory.

If no matching file is found, the parent and parent sibling directories of the current directory are scanned in an alphabetical breadth-first traversal to a height of one parent directory.
```

Files matching the pattern are processed in the order encountered. When a Music Manifest is encountered, it is opened and scanned for an MPV Album (Presentation Profile) or AssetList. The first MPV Album encountered is used for presentation; if none is found, the AssetList is used.

The rationale behind this search algorithm is to first locate any top-level manifest containing MPV information, with a fallback of then finding named Music Manifests. It is allowed for the MPV document to be located up to one directory down.

N.B. By allowing the Music Manifest to carry the .XML extension or type, general purpose XML processors can operate on the MPV document and apply XML processing capabilities. For example, with Microsoft Internet Explorer 5.5 and above, an XML processing instruction in the ALBUMMUM.XML file can invoke a style sheet that can transform the MPV document into an attractive browser-based presentation.

The search algorithm covers all of the following directories, where CWD is the current working directory. Naturally, when the path cannot be reached, it stops.

```
/R1/P1/CWD
/R1/P1/CWD/C1
/R1/P1/CWD/C2
```
In each of the directories scanned, the application shall search for all of the possible Music Manifest file names.

6.5 Media Types and File Formats

MPV is an open asset management / playlist format that can support an expandable set of defined media file formats. Formats are identified using MIME media types, as is well-established practice for internet-era standards.

The [MPV-Core] specification defines the following music audio formats.

**MEDIA TYPES FOR MPV:AUDIO FROM [MPV-CORE]**

<table>
<thead>
<tr>
<th>MIME Media Type</th>
<th>Mac File Type</th>
<th>PC File Suffixes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>audio/basic</td>
<td>ULAW</td>
<td>au, snd</td>
<td>8K, mono audio</td>
</tr>
<tr>
<td>audio/midi</td>
<td>MIDI</td>
<td>mid, midi</td>
<td>Musical Instrument Digital Interface sound file</td>
</tr>
<tr>
<td>audio/mpeg</td>
<td>MPEG</td>
<td>mp1, mp2, mp3, mpg, mpeg, m2a, m3a</td>
<td>MPEG audio layers 1, 2, and 3 as defined by [MIMEMPEG]</td>
</tr>
<tr>
<td>audio/wav</td>
<td>WAVE</td>
<td>wav</td>
<td>WAVE file</td>
</tr>
<tr>
<td>audio/x-aiff</td>
<td>AIFF</td>
<td>aif, aiff</td>
<td>Audio interchange file format</td>
</tr>
<tr>
<td>audio/x-ms-wma</td>
<td></td>
<td>wma</td>
<td>Windows Media Audio</td>
</tr>
</tbody>
</table>

This MPV Music Profile specification defines the following additional music audio formats.

**ADDITIONAL MEDIA TYPES FOR MPV:AUDIO**

<table>
<thead>
<tr>
<th>MIME Media Type</th>
<th>Mac File Type</th>
<th>PC File Suffixes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>audio/ac3</td>
<td></td>
<td>ac3</td>
<td>Dolby Digital 5.1 audio used widely for DVD-Video</td>
</tr>
<tr>
<td>audio/MP4A-LATM</td>
<td></td>
<td>aac, m4a, mp4, mpeg4</td>
<td>Audio using MPEG4 AAC with Low-overhead Audio Transport Multiplex as defined in [MIMEMPEG4]</td>
</tr>
<tr>
<td>audio/mpa</td>
<td>MPEG</td>
<td>mpa</td>
<td>MPEG1 or 2 audio sent by RTP as defined by [MIMEMPA]</td>
</tr>
<tr>
<td>????</td>
<td></td>
<td>.omg</td>
<td>Sony’s ATRAC3 format</td>
</tr>
</tbody>
</table>

**MEDIA TYPES FOR MPV:DOCUMENT**

The MPV Music Profile can support links to other playlists using the mpv:Document asset type and an appropriate MIME type entry in nmf:Metadata|dc:Properties|dc:format.
6.6 Embedded Media Types for MPV Music Profile

6.6.1 Audio File with ID3v2 Tagged Embedded Still Picture

The MPV Core specification [MPVCore] defines embedded media types in Appendix I.1. An embedded media type defines a means for referencing an asset that is embedded in a container format. The approach utilized is consistent with URL construction, namely to define a fragment identifier syntax for a specific container media type.

The ID3v2 specifications [ID3-230][ID3-240] supported still pictures embedded in MPEG1 and MPEG2 audio files and other file types. The pictures can be used for artwork. In contrast, the MPV Music Profile represents artwork as a discrete asset related to the music’s audio asset. It is required for the MPV Music Profile to be able to reference still picture content embedded in MPEG1 and MPEG2 files conformant to the ID3v2 specification.

To do so, the MPV Music Profile defines a new embedded media type and fragment identifier syntax.

MPV defines two embedded media types that are shown below for ID3v2 [ID3-230, ID3-240] embedded pictures.

**ID3v2 Embedded Media Type for MPV:Still Pictures**

Any of the MIME-types that can be used with an MPV:Still asset can also be used with an MPV:Still asset that is embedded in the APIC frame of an ID3v2 compliant audio file. The standard transformation of the MIME-type the unembedded MPV:Still should be used to arrive at the media type for the embedded still (see Appendix III:).

As an example if the Still being embedded in an APIC frame is of type png, then the MIME Media type would be:

```
Image/vnd.osta-org.emb.png
```

**Fragment Identifier Syntax for ID3v2 Embedded Media Type**

The Fragment Identifier is the text that follows the ‘#’ character in a URL, as in

```
"<base URL>#<fragment identifier>"
```

The fragment identifier syntax for ID3v2 uses a two part embedding specific locator, where the first part specifies the frame type that the embedded asset is located in and the second part specifies the frame specific information such as the picture type in the case of the APIC frame. The APIC frame is the only frame type that is used in the current version of the Music Profile.

<table>
<thead>
<tr>
<th>Fragment Identifier Syntax</th>
<th>Picture Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>#vnd.osta-org.emb/ID3v2(APIC, &lt;pictureType&gt;)</td>
<td>The following &lt;pictureType&gt; values are defined by ID3v2:</td>
</tr>
<tr>
<td></td>
<td>$00  Other</td>
</tr>
<tr>
<td></td>
<td>$01  32x32 pixels 'file icon' (PNG only)</td>
</tr>
</tbody>
</table>
Only one embedded picture of each PictureType is allowed in a ID3v2-conformant audio file.

<table>
<thead>
<tr>
<th>Example URLs</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$00)</td>
<td>Other file icon</td>
</tr>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$01)</td>
<td>Cover (front)</td>
</tr>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$02)</td>
<td>Cover (back)</td>
</tr>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$03)</td>
<td>Leaflet page</td>
</tr>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$04)</td>
<td>Media (e.g. label side of CD)</td>
</tr>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$05)</td>
<td>Lead artist/lead performer/soloist</td>
</tr>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$06)</td>
<td>Artist/performer</td>
</tr>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$07)</td>
<td>Conductor</td>
</tr>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$08)</td>
<td>Band/Orchestra</td>
</tr>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$09)</td>
<td>Composer</td>
</tr>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$0A)</td>
<td>Lyricist/text writer</td>
</tr>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$0B)</td>
<td>Recording Location</td>
</tr>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$0C)</td>
<td>During recording</td>
</tr>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$0D)</td>
<td>During performance</td>
</tr>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$0E)</td>
<td>Movie/video screen capture</td>
</tr>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$0F)</td>
<td>A bright coloured fish</td>
</tr>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$10)</td>
<td>Illustration</td>
</tr>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$11)</td>
<td>Band/artist logotype</td>
</tr>
<tr>
<td>music.mp3#vnd.osta-org.emb/ID3v2(APIC,$12)</td>
<td>Publisher/Studio logotype</td>
</tr>
</tbody>
</table>

**EXAMPLE**

```xml
<mpv:AssetList>
  <mpv:Audio mpv:id="ID000001">
    <mpv:LastURL>music.mp3</mpv:LastURL>
    <nmf:Metadata>
      <dc:Properties>
        <dc:format>audio/mpeg</dc:format>
      </dc:Properties>
  </mpv:Audio>
</mpv:AssetList>
```
Chapter 7: MPV Music Profile Mapping To Other Music Metadata Formats

7.1 ID3 and OSTA MPV Music Profile

The ID3 specifications are popular metadata representations for music. More information can be found in [ID3-230] and [ID3-240]. The OSTA MPV Music Profile specification provides similar capabilities within the context of the XML-based MPV specification framework.

The following mapping table can be used to associate ID3V1.0 and V1.1 terms and concepts with MPV Music Profile terms and concepts. ID3v2.0 provides much more extensive metadata and is not supported with the MPV Music Profile 1.0.

<table>
<thead>
<tr>
<th>ID3</th>
<th>MPV Music Profile</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID3v1</td>
<td>All specified under mpv:Audio</td>
<td>nmf:Metadata</td>
</tr>
<tr>
<td>Song title</td>
<td>dc:Properties</td>
<td>dc:title</td>
</tr>
<tr>
<td>Artist</td>
<td>dc:Properties</td>
<td>dc:creator</td>
</tr>
<tr>
<td>Album</td>
<td>mpvm:MusicProperties:AlbumTitle</td>
<td></td>
</tr>
<tr>
<td>Year</td>
<td>mpvm:MusicProperties:Recorded</td>
<td></td>
</tr>
<tr>
<td>Comment</td>
<td>dc:Properties</td>
<td>dc:description</td>
</tr>
<tr>
<td>Genre</td>
<td>mpvm:MusicProperties:Genre</td>
<td></td>
</tr>
</tbody>
</table>

| ID3v1.1   | All specified under mpv:Audio|nmf:Metadata                                                              |
| Song title     | dc:Properties|dc:title                                                               |
| Artist    | dc:Properties|dc:creator                                                             |
| Album     | mpvm:MusicProperties:AlbumTitle                |
| Year      | mpvm:MusicProperties:Recorded                  |
| Comment   | dc:Properties|dc:description                                                         |
| Album Track   | mpvm:MusicProperties:TrackNumber               |
| Genre     | mpvm:MusicProperties:Genre                      |

| ID3v2.3   | All specified under                      |

Ver. 0.90WD, 21 July 2003 © 2002-2003 OSTA 58 of 77
### ID3v2 Frame, Name

<p>| ID3v2 Frame, Name                  | V2.3 | V2.4 | All specified under <code>mpv:Audio|nmf:Metadata</code> unless noted |
|------------------------------------|------|------|-------------------------|
| AENC, Audio encryption            | X    | X    |                         |
| APIC, Attached picture            | X    | X    |                         |
| ASPI, Audio seek point index      | -    | X    | The <code>&lt;mpv:Still&gt;</code> referenced by <code>mpv:Audio|mpv:Related|mpv:StillRef</code>, where the <code>mpv:relationship</code> is one of the artwork identifiers |
| COMM, Comments                    | X    | X    | dc:Properties|dc:description |
| EQUA, Equalisation                | X    | Use EQU2 |
| EQU2, Equalisation (2)            | -    | X    |                         |
| ENCR, Encryption method registration | X | X |                         |
| ETCO, Event timing codes          | X    | X    |                         |
| GEOB, General encapsulated object | X    | X    | Asset referenced by <code>mpv:Audio|mpv:Related|mpv:&lt;assettype&gt;Ref</code> |
| GRID, Group identification registration | X | X |                         |
| IPLS, Involved people list        | X    | Use TMCL, TIPL |
| LINK, Linked information          | X    | X    |                         |
| MCDI, Music CD identifier         | X    | X    | <code>mpv:Audio|mpv:ContentID</code> where the <code>ContentID</code> is urn:osta-org:mpv:dsig:mcdi:&lt;mcdi value in hex&gt;:&lt;track number&gt; |
| MLLT, MPEG location lookup table  | X    | X    |                         |
| OWNE, Ownership frame             | X    | X    |                         |
| PRIV, Private frame               | X    | X    | use <code>mpv:Metadata</code> or <code>nmf:Metadata</code> to hold user data conforming to their own schema |
| PCNT, Play counter                | X    | X    |                         |
| POPM, Popularimeter               | X    | X    |                         |
| POSS, Position synchronisation frame | X | X |                         |
| RBUF, Recommended buffer size     | X    | X    |                         |
| RVAD, Relative volume adjustment  | X    | Use RVA2 |
| RVA2, Relative volume adjustment (2) | - | X |                         |
| RVRB, Reverb                      | X    | X    |                         |
| SEEK, Seek frame                  | -    | X    |                         |
| SIGN, Signature frame             | -    | X    |                         |
| SYLT, Synchronised lyric/text     | X    | X    | <code>mpv:MusicProperties|mpv:Lyrics</code> |
| SYTC, Synchronised tempo codes    | X    | X    |                         |
| TALB, Album/Movie/Show title      | X    | X    | <code>mpv:MusicProperties|AlbumTitle</code> |
| TBPM, BPM (beats per minute)      | X    | X    |                         |</p>
<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
<th>Present</th>
<th>Use</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCOM, X</td>
<td>Composer</td>
<td>X</td>
<td></td>
<td>mpvm:MusicProperties/Genre</td>
</tr>
<tr>
<td>TCON, X</td>
<td>Content type</td>
<td>X</td>
<td></td>
<td>dc:Properties/dc:rights</td>
</tr>
<tr>
<td>TCOP, X</td>
<td>Copyright message</td>
<td>X</td>
<td></td>
<td>mpvm:MusicProperties:Recorded</td>
</tr>
<tr>
<td>TDAT, X</td>
<td>Date</td>
<td>Use TDRC</td>
<td></td>
<td>mpvm:MusicProperties:Recorded</td>
</tr>
<tr>
<td>TDEN, X</td>
<td>Encoding time</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDLY, X</td>
<td>Playlist delay</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDOR, X</td>
<td>Original release time</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDRL, X</td>
<td>Recording time</td>
<td>-</td>
<td></td>
<td>dcterms:Properties:dcterms:issued</td>
</tr>
<tr>
<td>TDG, X</td>
<td>Tagging time</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TENC, X</td>
<td>Encoded by</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TEXT, X</td>
<td>Lyricist/Text writer</td>
<td>X</td>
<td></td>
<td>mpvm:MusicProperties/LyricsBy</td>
</tr>
<tr>
<td>TFLT, X</td>
<td>File type</td>
<td>X</td>
<td></td>
<td>dc:Properties/dc:format</td>
</tr>
<tr>
<td>TIME, X</td>
<td>Time</td>
<td>Use TDRC</td>
<td></td>
<td>mpvm:MusicProperties:Recorded</td>
</tr>
<tr>
<td>TIPL, X</td>
<td>Involved people list</td>
<td>-</td>
<td></td>
<td>mpvm:MusicProperties/LyricsBy or MusicBy or ArrangedBy</td>
</tr>
<tr>
<td>TIT1, X</td>
<td>Content group description</td>
<td>X</td>
<td></td>
<td>dc:Properties/dc:title</td>
</tr>
<tr>
<td>TIT2, X</td>
<td>Title/songname/content description</td>
<td>X</td>
<td></td>
<td>dc:Properties/dc:description</td>
</tr>
<tr>
<td>TIT3, X</td>
<td>Subtitle/Description refinement</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TKEY, X</td>
<td>Initial key</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TLAN, X</td>
<td>Language(s)</td>
<td>X</td>
<td></td>
<td>dc:Properties/dc:language</td>
</tr>
<tr>
<td>TLEN, X</td>
<td>Length</td>
<td>X</td>
<td></td>
<td>mpvm:MusicProperties/PlayingTime (floating point value in seconds)</td>
</tr>
<tr>
<td>TMCL, X</td>
<td>Musician credits list</td>
<td>-</td>
<td></td>
<td>mpvm:MusicProperties/PerformedBy</td>
</tr>
<tr>
<td>TMED, X</td>
<td>Media type</td>
<td>X</td>
<td></td>
<td>dcterms:Properties:dcterms:medium</td>
</tr>
<tr>
<td>TMOO, X</td>
<td>Mood</td>
<td>-</td>
<td></td>
<td>mpvm:MusicProperties/Mood</td>
</tr>
<tr>
<td>TOAL, X</td>
<td>Original album/movie/show title</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOFN, X</td>
<td>Original filename</td>
<td>X</td>
<td></td>
<td>Value of mpv:Audio/mpv:LastURL with the mpv:filesystem set to the filesystem of the original filename</td>
</tr>
<tr>
<td>TOLY, X</td>
<td>Original lyricist(s)/text writer(s)</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOPE, X</td>
<td>Original artist(s)/performer(s)</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TORY, X</td>
<td>Original release year</td>
<td>Use TDOR</td>
<td></td>
<td>tdor:Properties/tdor:year</td>
</tr>
<tr>
<td>TOWN, X</td>
<td>File owner/licensee</td>
<td>X</td>
<td></td>
<td>mpvm:MusicProperties:PrincipalArtist</td>
</tr>
<tr>
<td>TPE1, X</td>
<td>Lead performer(s)/Soloist(s)</td>
<td>X</td>
<td></td>
<td>Comment: dc:creator MUST always be provided. It can be same as PrincipalArtist or more complete.</td>
</tr>
<tr>
<td>TPE2, X</td>
<td>Band/orchestra/accompaniment</td>
<td>X</td>
<td></td>
<td>dc:Properties/dc:contributor</td>
</tr>
<tr>
<td>TPE3, X</td>
<td>Conductor/performer refinement</td>
<td>X</td>
<td></td>
<td>dc:Properties/dc:contributor</td>
</tr>
<tr>
<td>TPE4, X</td>
<td>Interpreted, remixed, or otherwise modified by</td>
<td>X</td>
<td></td>
<td>dc:Properties/dc:contributor</td>
</tr>
<tr>
<td>TPOS, X</td>
<td>Part of a set/Total number sets</td>
<td>X</td>
<td></td>
<td>mpvm:MusicProperties:SetNumber and mpvm:MusicProperties:NumSets</td>
</tr>
<tr>
<td>TPRO, X</td>
<td>Produced notice</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TPUB, X</td>
<td>Publisher</td>
<td>X</td>
<td></td>
<td>dc:Properties/dc:publisher</td>
</tr>
<tr>
<td>TRCK, X</td>
<td>Track number/Position in set</td>
<td>X</td>
<td></td>
<td>mpvm:MusicProperties:TrackNumber and mpvm:MusicProperties:NumTracks</td>
</tr>
<tr>
<td>TRDA, X</td>
<td>Recording dates</td>
<td>Use TDRC</td>
<td></td>
<td>mpvm:MusicProperties:Recorded</td>
</tr>
<tr>
<td>TRSN, X</td>
<td>Internet radio station name</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRSO, X</td>
<td>Internet radio station owner</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
7.1.1 Genre Mapping

ID3v1 [ID3] defines a set of genre category names and associated genre identifier numbers. MPV Music Profile makes use of the same genre names; numbers are not used. However, the Genre are listed in the Music Profile XML Schema in the same order as the ID3v1 genre are defined, make mapping straight forward for implementers.
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Oldies</td>
</tr>
<tr>
<td>12</td>
<td>Other</td>
</tr>
<tr>
<td>13</td>
<td>Pop</td>
</tr>
<tr>
<td>14</td>
<td>R&amp;B</td>
</tr>
<tr>
<td>15</td>
<td>Rap</td>
</tr>
<tr>
<td>16</td>
<td>Reggae</td>
</tr>
<tr>
<td>17</td>
<td>Rock</td>
</tr>
<tr>
<td>18</td>
<td>Techno</td>
</tr>
<tr>
<td>19</td>
<td>Industrial</td>
</tr>
<tr>
<td>20</td>
<td>Alternative</td>
</tr>
<tr>
<td>21</td>
<td>Ska</td>
</tr>
<tr>
<td>22</td>
<td>Death Metal</td>
</tr>
<tr>
<td>23</td>
<td>Pranks</td>
</tr>
<tr>
<td>24</td>
<td>Soundtrack</td>
</tr>
<tr>
<td>25</td>
<td>Euro-Techno</td>
</tr>
<tr>
<td>26</td>
<td>Ambient</td>
</tr>
<tr>
<td>27</td>
<td>Trip-Hop</td>
</tr>
<tr>
<td>28</td>
<td>Vocal</td>
</tr>
<tr>
<td>29</td>
<td>Jazz+Funk</td>
</tr>
<tr>
<td>30</td>
<td>Fusion</td>
</tr>
<tr>
<td>31</td>
<td>Trance</td>
</tr>
<tr>
<td>32</td>
<td>Classical</td>
</tr>
<tr>
<td>33</td>
<td>Instrumental</td>
</tr>
<tr>
<td>34</td>
<td>Acid</td>
</tr>
<tr>
<td>35</td>
<td>House</td>
</tr>
<tr>
<td>36</td>
<td>Game</td>
</tr>
<tr>
<td>37</td>
<td>Sound Clip</td>
</tr>
<tr>
<td>38</td>
<td>Gospel</td>
</tr>
<tr>
<td>39</td>
<td>Noise</td>
</tr>
<tr>
<td>40</td>
<td>AlternRock</td>
</tr>
<tr>
<td>41</td>
<td>Bass</td>
</tr>
<tr>
<td>42</td>
<td>Soul</td>
</tr>
<tr>
<td>43</td>
<td>Punk</td>
</tr>
<tr>
<td>44</td>
<td>Space</td>
</tr>
<tr>
<td>45</td>
<td>Meditative</td>
</tr>
<tr>
<td>46</td>
<td>Instrumental Pop</td>
</tr>
<tr>
<td>47</td>
<td>Instrumental Rock</td>
</tr>
<tr>
<td>48</td>
<td>Ethnic</td>
</tr>
<tr>
<td>49</td>
<td>Gothic</td>
</tr>
<tr>
<td>50</td>
<td>Darkwave</td>
</tr>
<tr>
<td>51</td>
<td>Techno-Industrial</td>
</tr>
<tr>
<td>52</td>
<td>Electronic</td>
</tr>
<tr>
<td>53</td>
<td>Pop-Folk</td>
</tr>
<tr>
<td>54</td>
<td>Eurodance</td>
</tr>
<tr>
<td>55</td>
<td>Dream</td>
</tr>
<tr>
<td>56</td>
<td>Southern Rock</td>
</tr>
<tr>
<td>57</td>
<td>Comedy</td>
</tr>
<tr>
<td>58</td>
<td>Cult</td>
</tr>
<tr>
<td>59</td>
<td>Gangsta</td>
</tr>
<tr>
<td>60</td>
<td>Top 40</td>
</tr>
<tr>
<td>61</td>
<td>Christian Rap</td>
</tr>
<tr>
<td>62</td>
<td>Pop/Funk</td>
</tr>
<tr>
<td>63</td>
<td>Jungle</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>64</td>
<td>Native American</td>
</tr>
<tr>
<td>65</td>
<td>Cabaret</td>
</tr>
<tr>
<td>66</td>
<td>New Wave</td>
</tr>
<tr>
<td>67</td>
<td>Psychadelic</td>
</tr>
<tr>
<td>68</td>
<td>Rave</td>
</tr>
<tr>
<td>69</td>
<td>Showtunes</td>
</tr>
<tr>
<td>70</td>
<td>Trailer</td>
</tr>
<tr>
<td>71</td>
<td>Lo-Fi</td>
</tr>
<tr>
<td>72</td>
<td>Tribal</td>
</tr>
<tr>
<td>73</td>
<td>Acid Punk</td>
</tr>
<tr>
<td>74</td>
<td>Acid Jazz</td>
</tr>
<tr>
<td>75</td>
<td>Polka</td>
</tr>
<tr>
<td>76</td>
<td>Retro</td>
</tr>
<tr>
<td>77</td>
<td>Musical</td>
</tr>
<tr>
<td>78</td>
<td>Rock &amp; Roll</td>
</tr>
<tr>
<td>79</td>
<td>Hard Rock</td>
</tr>
<tr>
<td>80</td>
<td>Folk</td>
</tr>
<tr>
<td>81</td>
<td>Folk-Rock</td>
</tr>
<tr>
<td>82</td>
<td>National Folk</td>
</tr>
<tr>
<td>83</td>
<td>Swing</td>
</tr>
<tr>
<td>84</td>
<td>Fast Fusion</td>
</tr>
<tr>
<td>85</td>
<td>Bebob</td>
</tr>
<tr>
<td>86</td>
<td>Latin</td>
</tr>
<tr>
<td>87</td>
<td>Revival</td>
</tr>
<tr>
<td>88</td>
<td>Celtic</td>
</tr>
<tr>
<td>89</td>
<td>Bluegrass</td>
</tr>
<tr>
<td>90</td>
<td>Avantgarde</td>
</tr>
<tr>
<td>91</td>
<td>Gothic Rock</td>
</tr>
<tr>
<td>92</td>
<td>Progressive Rock</td>
</tr>
<tr>
<td>93</td>
<td>Psychedelic Rock</td>
</tr>
<tr>
<td>94</td>
<td>Symphonic Rock</td>
</tr>
<tr>
<td>95</td>
<td>Slow Rock</td>
</tr>
<tr>
<td>96</td>
<td>Big Band</td>
</tr>
<tr>
<td>97</td>
<td>Chorus</td>
</tr>
<tr>
<td>98</td>
<td>Easy Listening</td>
</tr>
<tr>
<td>99</td>
<td>Acoustic</td>
</tr>
<tr>
<td>100</td>
<td>Humour</td>
</tr>
<tr>
<td>101</td>
<td>Speech</td>
</tr>
<tr>
<td>102</td>
<td>Chanson</td>
</tr>
<tr>
<td>103</td>
<td>Opera</td>
</tr>
<tr>
<td>104</td>
<td>Chamber Music</td>
</tr>
<tr>
<td>105</td>
<td>Sonata</td>
</tr>
<tr>
<td>106</td>
<td>Symphony</td>
</tr>
<tr>
<td>107</td>
<td>Booty Bass</td>
</tr>
<tr>
<td>108</td>
<td>Primus</td>
</tr>
<tr>
<td>109</td>
<td>Porn Groove</td>
</tr>
<tr>
<td>110</td>
<td>Satire</td>
</tr>
<tr>
<td>111</td>
<td>Slow Jam</td>
</tr>
<tr>
<td>112</td>
<td>Club</td>
</tr>
<tr>
<td>113</td>
<td>Tango</td>
</tr>
<tr>
<td>114</td>
<td>Samba</td>
</tr>
<tr>
<td>115</td>
<td>Folklore</td>
</tr>
<tr>
<td>116</td>
<td>Ballad</td>
</tr>
</tbody>
</table>
7.2 WinAMP M3U and OSTA MPV Music Profile

The WinAMP M3U playlist is commonly encountered. The following illustrates mapping M3U playlist to the MPV Music Profile.

<table>
<thead>
<tr>
<th>M3U Playlist</th>
<th>MPV Music Profile</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Song title</td>
<td>mpv:Audio</td>
<td>nmf:Metadata</td>
</tr>
<tr>
<td>Filename</td>
<td>mpv:Audio</td>
<td>mpv:LastURL</td>
</tr>
<tr>
<td>Duration</td>
<td>mpvm:MusicProperties</td>
<td>PlayingTime</td>
</tr>
</tbody>
</table>

7.3 OSTA MultiAudio and OSTA MPV Music Profile

The MultiAudio specification, already developed by OSTA [MultiAudio], is an earlier generation of technology specific to audio. MPV integrates music, photos, and video and can be used to create and exchange multimedia playlists and collections. Implementers are encouraged to utilize MPV.

The OSTA MultiAudio specification provides a CD or DVD table of contents and playlist representation for compressed audio content on data discs. This binary format is suitable for implementation in very resource-constrained devices.

The OSTA MPV Music Profile specification provides similar capabilities within the context of the XML-based MPV specification framework. This allows a single consistent multimedia album format to span music, photo, and video content. For consumer electronics devices able to provide an implementation of the MPV framework, the MPV Music Profile offers a means to support all multimedia content within a consistent framework and single firmware implementation.

The following mapping table can be used to associate MultiAudio terms and concepts with MPV Music Profile terms and concepts:

<table>
<thead>
<tr>
<th>MultiAudio</th>
<th>MPV Music Profile</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrackEntry</td>
<td>mpv:Audio asset</td>
<td></td>
</tr>
<tr>
<td>Pathname</td>
<td>mpv:Audio</td>
<td>mpv:LastURL</td>
</tr>
<tr>
<td>Track Name</td>
<td>dc:Properties</td>
<td>dc:title</td>
</tr>
<tr>
<td>Performer Name</td>
<td>dc:Properties</td>
<td>dc:creator</td>
</tr>
<tr>
<td>Property</td>
<td>Namespace and Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Composer Name</td>
<td>mpvm:MusicProperties:MusicBy</td>
<td></td>
</tr>
<tr>
<td>Songwriter Name</td>
<td>mpvm:MusicProperties:LyricsBy</td>
<td></td>
</tr>
<tr>
<td>Arranger Name</td>
<td>mpvm:MusicProperties:ArrangedBy</td>
<td></td>
</tr>
<tr>
<td>Album Name</td>
<td>mpvm:MusicProperties:AlbumTitle</td>
<td></td>
</tr>
<tr>
<td>Genre</td>
<td>mpvm:MusicProperties:Genre</td>
<td></td>
</tr>
<tr>
<td>Playing Time</td>
<td>mpvm:MusicProperties:PlayingTime</td>
<td></td>
</tr>
<tr>
<td>Year Recorded</td>
<td>mpvm:MusicProperties:Recorded</td>
<td></td>
</tr>
<tr>
<td>Track Order</td>
<td>mpvm:MusicProperties:TrackNumber</td>
<td></td>
</tr>
<tr>
<td>Number of Channels</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average Encoded Bitrate</td>
<td>mpvm:MusicProperties:EncodedBitrate</td>
<td>Use average bitrate for CBR encoded assets.</td>
</tr>
<tr>
<td>Maximum Bitrate</td>
<td>mpvm:MusicProperties:EncodedBitrate</td>
<td>Use maximum bitrate for VBR encoded assets.</td>
</tr>
<tr>
<td>Sample Rate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extra Data</td>
<td>mpv:Metadata and nmf:Metadata</td>
<td></td>
</tr>
<tr>
<td><strong>Playlist</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Tracks</td>
<td>mpvm:MusicProperties:NumTracks</td>
<td></td>
</tr>
<tr>
<td>Playlist Name</td>
<td>dc:Properties/dc:title</td>
<td></td>
</tr>
<tr>
<td>Playlist Description</td>
<td>dc:Properties/dc:description</td>
<td></td>
</tr>
<tr>
<td>Track Indexes</td>
<td>mpvp:Album:Foreground contents</td>
<td></td>
</tr>
<tr>
<td>Extra Data</td>
<td>mpv:Metadata and nmf:Metadata</td>
<td></td>
</tr>
<tr>
<td><strong>Playlist Directory</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>dc:Properties/dc:title</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>dc:Properties/dc:description</td>
<td></td>
</tr>
<tr>
<td>Tracklist Pathnames</td>
<td>mpvp:AlbumRef or mpv:ManifestLinkRef</td>
<td>in the mpvp:Album:Foreground</td>
</tr>
<tr>
<td>Playlist Indexes</td>
<td>mpvp:AlbumRef or mpv:ManifestLinkRef</td>
<td>in the mpvp:Album:Foreground</td>
</tr>
<tr>
<td>Extra Data</td>
<td>mpv:Metadata and nmf:Metadata</td>
<td></td>
</tr>
<tr>
<td><strong>TOC_HEADER</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version Number</td>
<td>encoded into profile and namespace identifiers</td>
<td></td>
</tr>
<tr>
<td>UUID</td>
<td>mpvId:InstanceID</td>
<td></td>
</tr>
<tr>
<td>Volume Name</td>
<td>dc:Properties/dc:title</td>
<td></td>
</tr>
<tr>
<td>Data Preparer Identifier</td>
<td>dc:Properties/dc:Publisher</td>
<td></td>
</tr>
<tr>
<td>Publisher Identifier</td>
<td>dc:Properties/dc:rights</td>
<td></td>
</tr>
<tr>
<td>Creation Date and Time</td>
<td>dcterms:Properties</td>
<td>dcterms:created</td>
</tr>
<tr>
<td>Modification Date and Time</td>
<td>dcterms:Properties</td>
<td>dcterms:modified</td>
</tr>
<tr>
<td>Effective Date and Time</td>
<td>dcterms:Properties</td>
<td>dcterms:issued</td>
</tr>
<tr>
<td>Expiration Date and Time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of Playlist Directories</td>
<td>Implicit</td>
<td></td>
</tr>
<tr>
<td>Number of Tracks</td>
<td>mpvm:MusicProperties:NumTracks</td>
<td></td>
</tr>
<tr>
<td>Number of Playlists</td>
<td>Implicit</td>
<td></td>
</tr>
<tr>
<td>Extra Data</td>
<td>mpv:Metadata and nmf:Metadata</td>
<td></td>
</tr>
</tbody>
</table>
Appendix I: References

[CSS2]

[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]

[DCF-1999]

[DIG35-2001]

[FreeDB]

[ID3]

[ID3-230]
[ID3-240]
“ID3 Tag Version 2.4.0 - Native Frames”, Nov. 1, 2000, Martin Nielson, an informal standard.
http://www.id3.org/id3v2.4.0-frames.txt
[ISO-639-2]

[ISO639-1]

[ISO3166-1]
‘ISO 3166-1 Codes for the representation of names of countries and their subdivisions - Part 1: Country codes.’

[ISO8601]

[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]
Available at http://www.w3.org/Graphics/JPEG/jfif.txt

[MANIFEST]
"XML Manifest Specification 1.0"; OSTA, 2002.,
Available at http://www.osta.org/mpv/

[MD5]

[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

[MIMEMPA]

[MIMEMPEG]
[MIME MPEG4]
Available at ftp://ftp.isi.edu/in-notes/rfc3016.txt

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

[MPV-Basic]
“MPV – Basic Profile Specification”, OSTA, 2002,
Available at http://www.osta.org/mpv/

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

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

[MultiAudio]
"MultiAudio 1.0"; OSTA, 2001.,
Available at http://www.osta.org/

[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]

[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]
W3C Recommendation 22 February 1999,
Available at http://www.w3.org/TR/REC-rdf-syntax/

[RDFschema]
Available at http://www.w3.org/TR/PR-rdf-schema/
[RFC1766]
Available at ftp://ftp.isi.edu/in-notes/rfc1766.txt

[SMIL10]
"Synchronized Multimedia Integration Language (SMIL) 1.0" P. Hoschka. W3C Recommendation 15 June 1998,
Available at http://www.w3.org/TR/REC-smil.

[SMIL20]
Available at http://www.w3.org/TR/smil20/

[SMIL-MOD]
"Synchronized Multimedia Modules based upon SMIL 1.0", Patrick Schmitz, Ted Wugofski and Warner ten Kate. W3C Note 23 February 1999,
Available at http://www.w3.org/TR/N ode-SY MM-modules

[URI]
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]

[W3C-NSURI]
Available at http://www.w3.org/1999/10/nsuri

[XML10]
"Extensible Markup Language (XML) 1.0" T. Bray, J. Paoli and C.M. Sperber-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

[XMP-FW]
"XMP – Extensible Metadata Platform 14 Sept 01", Copyright 2001 Adobe Inc,

[XSCHEMA]
Available at http://www.w3.org/TR/xmlschema-1/

[XSL]
Available at http://www.w3.org/TR/xsl/
Appendix II: Music CD Identifier

OSTA does not endorse the use of any specific music identification service. However, [ID3v240] defines a MusicCDIdentifier based on the CDBB ID calculating function developed by Ti Kan, ti@amb.org, for use in xmcd, http://amb.org/xmcd/ and used by [CDBB] and [FreeDB] and other music identification services to access information about music CDs. This description is excerpted from [FreeDB] and Copyright(c) 2003 FreeDB.org.

The disc ID is a 8-digit hexadecimal (base-16) number, computed using data from a CD's Table-of-Contents (TOC) in MSF (Minute Second Frame) form. The algorithm is listed below in Appendix A.

It is crucial that your software computes the disc ID correctly. If it does not generate the disc ID correctly, it will not be compatible with the freedb. Moreover, if your software submits freedb entries with bad disc IDs to the freedb archives, it could compromise the integrity of the freedb.

We suggest installing one of the disc ID generator programs listed on the freedb web page in the downloads/misc section, and then testing the disc ID code in your software by comparing the disc ID generated by the program with that of your software for as large a number of CDs as possible. Alternatively you can e.g. use xmcd and compare the DiscID generated by xmcd with that of your software. Bugs in disc ID calculation can be subtle, and history shows that it sometimes takes hundreds of discs to find problems.

APPENDIX A - CDBB/FREEDB DISCID ALGORITHM
-----------------------------------------

The following is a C code example that illustrates how to generate the CDBB/freedb disc ID. [...] A text description of the algorithm follows, which should contain the necessary information to code the algorithm in any programming language.

```c
struct toc {
    int min;
    int sec;
    int frame;
};

struct toc cdtoc[100];

int read_cdtoc_from_drive(void) {
    /* Do whatever is appropriate to read the TOC of the CD */
    /* into the cdtoc[] structure array. */
    return (tot_trks);
```

Ver. 0.90WD, 21 July 2003 © 2002-2003 OSTA 70 of 77
```c
int
cddb_sum(int n)
{
    int ret;

    /* For backward compatibility this algorithm must not change */
    ret = 0;
    while (n > 0) {
        ret = ret + (n % 10);
        n = n / 10;
    }
    return (ret);
}

unsigned long
cddb_discid(int tot_trks)
{
    int i,
    t = 0,
    n = 0;

    /* For backward compatibility this algorithm must not change */
    i = 0;
    while (i < tot_trks) {
        n = n + cddb_sum((cdtoc[i].min * 60) + cdtoc[i].sec);
        i++;
    }

    t = ((cdtoc[tot_trks].min * 60) + cdtoc[tot_trks].sec) -
        ((cdtoc[0].min * 60) + cdtoc[0].sec);

    return ((n % 0xff) << 24 | t << 8 | tot_trks);
}

int
main()
{
    int tot_trks;

    tot_trks = read_cdtoc_from_drive();
    printf("The discid is %08x", cddb_discid(tot_trks));
}
```

This code assumes that your compiler and architecture support 32-bit integers.

The `cddb_discid` function computes the discid based on the CD's TOC data in MSF form. The frames are ignored for this purpose. The function is passed a parameter of `tot_trks` (which is the total number of tracks on the CD), and returns the discid integer number.

It is assumed that `cdtoc[]` is an array of data structures (records) containing the fields `min`, `sec` and `frame`, which are the minute, second and frame offsets (the starting location) of each track. This information is read from the TOC of the CD. There are actually `tot_trks + 1` “active” elements in the array, the last one being the offset of the lead-out (also known as track 0xAA).

The function loops through each track in the TOC, and for each track it takes the `(M * 60) + S` (total offset in seconds) of the track and feeds it to `cddb_sum()` function, which simply adds the value of each digit in the decimal string representation of the number. A running sum of this result for each track is kept in the variable `n`.

At the end of the loop:
1. `t` is calculated by subtracting the `(M * 60) + S` offset of the lead-out minus
the \((M \times 60) + S\) offset of first track (yielding the length of the disc in seconds).

2. The result of \((n \ mod\ FFh)\) is left-shifted by 24 bits.

3. \(t\) is left shifted by 8.

The bitwise-OR operation of result 2., 3. and the \(\text{tot\_trks}\) number is used as the discid.

The discid is represented in hexadecimal form for the purpose of \(\text{xmcd\ cddb}\) file names and the \(\text{DISCID}^=\) field in the \(\text{xmcd\ cddb}\) file itself. If the hexadecimal string is less than 8 characters long, it is zero-padded to 8 characters (i.e., \(3a8f07\) becomes \(003a8f07\)). All alpha characters in the string should be in lower case, where applicable.

Important note for clients using the MS-Windows MCI interface:

The Windows MCI interface does not recognize data tracks, as you find them on CD Extra CD's. Therefore a wrong disc ID is generated for CD Extra's when using the MCI interface to read the TOC. Because of this, using the MCI interface should only be the last resort - if possible you should use other methods to read the TOC, like ASPI calls. An example disc ID calculator using ASPI can be found on the freedb website along with the sourcecode.

If for some reason, there is no other way for your program, than to use the MCI interface, here is the description how to do so:

The Windows MCI interface does not provide the MSF location of the lead-out. Thus, you must compute the lead-out location by taking the starting position of the last track and add the length of the last track to it. However, the MCI interface returns the length of the last track as ONE FRAME SHORT of the actual length found in the CD's TOC. In most cases this does not affect the disc ID generated, because we truncate the frame count when computing the disc ID anyway. However, if the lead-out track has an actual frame count of 0, the computed quantity (based on the MSF data returned from the MCI interface) would result in the seconds being one short and the frame count be 74. For example, a CD with the last track at an offset of 48m 32s 12f and having a track length of 2m 50s 63f has a lead-out offset of 51m 23s 0f long. Windows MCI incorrectly reports the length as 2m 50s 62f, which would yield a lead-out offset of 51m 22s 74f, which causes the resulting truncated disc length to be off by one second. This will cause an incorrect disc ID to be generated. You should thus add one frame to the length of the last track when computing the location of the lead-out.

The easiest way for Windows clients to compute the lead-out given information in MSF format is like this:

\[
(\text{offset\_minutes} \times 60 \times 75) + (\text{offset\_seconds} \times 75) + \text{offset\_frames} + \\
(\text{length\_minutes} \times 60 \times 75) + (\text{length\_seconds} \times 75) + \text{length\_frames} + 1 = \text{X}
\]

Where \(X\) is the offset of the lead-out in frames. To find the lead-out in seconds, simply divide by 75 and discard the remainder.
Appendix III: Identification of Embedded Assets

This appendix describes enhancements to section 5.8 and Appendix I.1 of the MPV Core specification [MPV-Core]. The general approach described in the MPV Core Specification is maintained with some modifications that allow better support for multi-level embedding and multiple embedding approaches for the same underlying asset type.

MPV supports the use of embedded media types using extensions to the MIME-types mechanism and URI fragment identifiers. MPV supports the ability to reference an embedded asset that is has more than one level of container wrapped around it. An example would be a ZIP file that contains an MP3 file that contains a picture of the artist that is embedded in the MP3 file using the approach defined in the ID3v2 specification.

**MEDIA TYPE SYNTAX**

The predefined embedded MIME-types are hosted in the OSTA namespace using its vendor root: “vnd.оста-org”. New embedded MIME-types can be defined using compatible mechanisms in any namespace that complies with MIME naming conventions; “vnd.оста-org” MIME types MUST be reserved for use by OSTA.

The general approach for embedded media-types is to name them using a simple transformation on the MIME-type that the assets would have if they weren’t embedded. As an example, if the non-embedded MIME-type of the JPEG thumbnail embedded in a JPEG file is “image/jpeg”, the embedded version would be “image/vnd.оста-org.emb.jpeg”.

The general transformation is to take the mime subtype the asset would have if it wasn’t embedded and prefix it with “vnd.оста-org.emb.” Some further examples:

- Image/tiff   ->   Image/vnd.оста-org.emb.tiff
- Audio/mp3   ->   Audio/vnd.оста-org.emb.mp3

**FRAGMENT IDENTIFIER SYNTAX**

The fragment identifier portion of a URL reference specifies what embedding schemes need to be processed in order to extract the embedded asset. The URL portion (the part before the “#”) identifies the outermost containing resource. The fragment identifier has two parts, the announcer part followed by one or more embedding scheme descriptors. There is an embedding scheme descriptor for each level of embedding that is present between the outermost resource and the asset that is being referenced.
The BNF below specifies the grammar for the fragment identifier used by MPV compliant embedded media types. This BNF conforms to the BNF usage from IETF specifications such as in [URI].

```
mpv-fragment = Announcer ["/"] scheme-descriptor]+
Announcer = (osta-root | other-root) ".emb"
Scheme descriptor = Container-type [ Locator ]
Container-type = This identifier must conform to the NCName production of XML.
osta-root = "vnd.osta-org"
Locator = (" embed-specific-location-scheme ")
embed-specific-location-scheme = <a well-formed locator >
```

If the locator contains parens, they must be balanced in order to allow the processor to recognize the end of the locator.

Below are the two embedded media type defined in the Core Specification as they would be specified using the new MPV defines two embedded media types that are shown below for Exif [Exif2002] embedded assets.

**EMBEDDING SCHEMES FOR MPV:STILL**

The MPV Core Specification defines a single embedding scheme for MPV:Still. The embedding scheme is defined in the Exif specification [Exif 2002]. The MIME Media Type for this embedding scheme is the same as for all other JPEG stills that are embedded in a container format, i.e.:

```
image/vnd.osta-org.emb.jpeg
```

The fragment identifier syntax is:

```
Exif-filename#vnd.osta-org.emb/EXIF(thumb)
```

**EMBEDDED MEDIA TYPES FOR MPV:AUDIO**

The MPV Core Specification defines a single embedding scheme for MPV:Audio. The embedding scheme is defined in the Exif specification [Exif 2002]. The MIME Media Type for this embedding scheme is the same as for all other wav audios that are embedded in a container format, i.e.:

```
image/vnd.osta-org.emb.wav
```

The fragment identifier syntax is:

```
Exif-filename#vnd.osta-org.emb/EXIF(audio)
```

In this example, the MPV document describes a single Exif file containing the original image, an embedded thumbnail, and an embedded audio. The fragment identifier syntax for the Exif thumbnail and audio is compliant with the usage described above.
<mpv:Still mpv:id="ID000101">
  <mpv:LastURL>IMG001.JPG</mpv:LastURL>
  <mpv:Rendition mpv:renditionUsage="thumbnail">
    <mpv:StillRef mpv:idRef="ID000103" />
  </mpv:Rendition>
</mpv:Still>

<mpv:Audio mpv:id="ID000102">
  <mpv:LastURL>IMG001.JPG#vnd.osta-org.emb.EXIF(audio)</mpv:LastURL>
  <nmf:Metadata>
    <dc:Properties>
      <dc:format>image/vnd.osta-org.emb.wav</dc:format>
    </dc:Properties>
  </nmf:Metadata>
</mpv:Audio>

<mpv:Still mpv:id="ID000103" mpv:instanceID="AC937BCFA3B340da971BAF09B17DBC324">
  <mpv:LastURL>IMG001.JPG#vnd.osta-org.emb.EXIF(thumb)</mpv:LastURL>
  <nmf:Metadata>
    <dc:Properties>
      <dc:format>image/vnd.osta-org.emb.jpeg</dc:format>
    </dc:Properties>
  </nmf:Metadata>
</mpv:Still>

...
Appendix IV: Changes

**VERSION 0.90**
- Changed the embedding approach to support more general embedding and nested embedding.
  Added an appendix with description of the new embedding approach.
- Added Executive summary

**VERSION 0.80**
Appendix V: Topics for Consideration & Comment

1. should MPV Music Profile recognize some other XML encoding of music properties or define its own. If some other, which and why.
2. should the Music Profile include metadata that captures “voting/rating” status as music properties?
3. should the MusicProfile include more metadata for use by online music services. If so, what?
4. Should we define MIME types and fragment accessors for various MPEG4 embedded content