A survey of playlist formats

Lucas Gonze <lucas@gonze.com>

November 17, 2003

This document is a survey of playlist data formats. It is useful in two ways. One, as a collation of data which is normally scattered all over the web, it is a helpful reference. Two, having this data in one place makes it easier to observe patterns.

Playlists are comparatively simple objects. They are nothing but lists -- here is the first song, here is the second. As a result they fail to excite the imagination of many people, because the expressive possibilities seem too limited. But from my background as a musician, arranger and composer, I know that the sequencing of aesthetic experiences has huge expressive possibilities. In my work on playlists I aim to help extend the expressive power of sequencing to objects on the world wide web.

Administrative Notes

Because I wanted the broadest possible view of the needs and abilities of applications that use playlists, I have attempted to make this listing exhaustive. As a result there are several hopelessly obscure entries, including one for which there is no real world implementation and one for which there is only a single instance in the real world.

The home of this document on the web is http://gonze.com/playlists/playlist-format-survey.html. The author of this document makes no claims and offers no warrantees.

I have quoted liberally from primary materials documenting individual playlist formats. I am grateful to the authors of the source documents.

This is an informal draft aimed at the community of people interested in playlists. Comments are welcome. I would be particularly grateful for corrections.

Observed Patterns

In this section I will comment on patterns observed among the different formats. This section precedes the facts on which it is based, so you may want to skip over to the reference before coming back here. I put it first because the reference section is dull stuff, not intended for browsing, and I doubt that any reader will discover this section if it is after the reference.

There are both syntactic and semantic differences. Syntactic differences include things like binary versus text formats, the number of dimensions a format can encompass, and so on. Semantic difference express application requirements, for example the issue of whether metadata about contained objects should be in the container or the object; this is an application requirement because its utility depends on the level of trust in the metadata, which varies from one usecase to another.

- *Dimensionality*: I have observed three levels of dimensionality in playlist formats. Flat files, which are the most popular, are one dimensional. INI files, which are based on a feature of the Windows API, are two dimensional. XML files support an arbitrary number of features, so are N-dimensional.

- *Mime types*: Mime types for playlist formats tend to be ad-hoc rather than formal; most have improvised names like application/x-something or text/fooCorp-barApp. There is a dedicated mime type for every playlist format. (Except for DAAP, which is intermingled with other data.)

- *Originators*: Few playlist formats are formally specified. Most are proprietary formats defined for compatibility with specific applications. One is an ad-hoc standard in the public domain with no

specification or owner, and one is a standard of the W3C.
- *Metadata*: Some playlist formats support metadata about contained objects, some don't. The dividing line is typically whether the data format supports more than one dimension. All of the XML formats support some form of metadata, but none use the Dublin Core standard.

- *Dependencies on referenced resources*: A number of formats are syntactically so similar that they can only be distinguished by the type of object they can reference. The RAM format, for example, is barely different from M3U, except that it can reference proprietary file types belonging to Real, which also owns the RAM format.

- *Resource locations*: Some formats expect resources to be remote URLs, some expect resources to be local files on the client, many allow remote resources grudgingly. iTunes, for example, will only play the first URL in an M3U file.

- *Application-defined catalogs versus shareable publications*: Design choices were usually made to satisfy the needs of a media-rendering application. Most playlist formats are catalogs of files stored on the client, with the ability to reference remote resources added as an afterthought if at all. As a result, these formats have as much in common with configuration files as they do with publication formats like HTML.

- *Complexity vs. simplicity*: There is wide divergence in the level of complexity. The most popular format, M3U, is also the simplest. In contrast, there is a noticeable scarcity of agents capable of using SMIL, which is one of the most complex formats.

- *Text rather than binary*: Out of all the following formats, only DAAP is binary. But the binary nature of DAAP is only syntactic sugar, since it is semantically identical to the textual XML format used by the iTunes library.

- *Orthogonal functions*: Playlist functionality encompasses audio metadata like the name of the artist; sequence information about the order and presence of songs; and presentation information like the background color of the player. Some formats mix these together, some keep them strictly separate. M3U specifies only the sequence, SMIL specifies only sequence and presentation, and MusicBrainz specifies only sequence and metadata.

Playlist Format Reference

Table of contents

Description: Every line in an M3U file is either a comment, a blank, or a resource to render. A comment line begins with the pound sign, #. Blanks are ignored. A resource is the address of a media file.

A resource address can be anything the M3U reader is capable of understanding. These include absolute filesystem paths, relative filesystem paths (with the base undefined by the file format), and URLs.

A resource can be anything the M3U reader is capable of rendering. To my knowledge these are always audio files, but there is no set reason for that to be true. However, it may not be wise to point to proprietary media formats like Real streaming audio in an M3U file, since many players will throw a user-visible error for media they cannot render.

The design philosophy of M3U is to let resource data types do the work. Players that don't understand an address or resource type usually skip the entry. The ability to reference URLs, in addition to filesystem paths, was added this way; some players (Winamp and XMMS, notably) simply added the ability to handle URLs to their M3U readers.

Support for M3U features varies wildly. iTunes, for example, will only render the first entry in an M3U file.

M3U is by far the most popular playlist format, probably due to its simplicity. It is an ad-hoc standard with no formal definition, no canonical source, and no owner.

Example:

```
# This is an absolute filesystem path
c:/music/foo.mp3

# This is a relative filesystem path
foo/fighters.mp3

# This is a URL
http://foofighters.com/somesong.mp3
```

Mime type: `audio/mpegurl (recommended)`

`audio/x-mpegurl`

Distinguishing features: A simple list of files, one per line.

Definition URL: http://www.schworak.com/programming/music/playlist_m3u.asp

Originator: Winamp (?)

Implementations: Winamp, XMMS, many more

Metadata support: Before ID3 tags were widely supported by MP3 players, a flavor of M3U called Extended M3U was used to indicate audio metadata. Extended M3U is now obsolete. The following description of Extended M3U is copied in verbatim from Google's cache of the reverse-engineered documentation at http://hanna.pyxidis.org/tech/m3u.html, which is now a defunct URL.

```
#EXTM3U
#EXTINF:111,3rd Bass - Al z A-B-Cee z
mp3/3rd Bass/3rd bass - Al z A-B-Cee z.mp3
#EXTINF:462,Apoptygma Berzerk - Kathy«s song (VNV Nation rmx)
mp3/Apoptygma Berzerk/Apoptygma Berzerk - Kathy's Song (Victoria Mix by VNV Nation).mp3
#EXTINF:394,Apoptygma Berzerk - Kathy's Song
mp3/Apoptygma Berzerk/Apoptygma Berzerk - Kathy's Song.mp3
#EXTINF:307,Apoptygma Bezerk - Starsign
mp3/Apoptygma Berzerk/Apoptygma Berzerk - Starsign.mp3
#EXTINF:282,Various_Artists - Butthole Surfers: They Came In
mp3/Butthole_Surfers-They_Came_In.mp3
```

The First line, "#EXTM3U" is the format descriptor, in this case M3U (or Extended M3U as it can be called). It does not change, it's always this.

The second and third operate in a pair. The second begins "#EXTINF:" which serves as the record marker. The "#EXTINF" is unchanging. After the colon is a number: this number is the length of the track in whole seconds (not minutes:seconds or anything else. Then comes a comma and the name of the tune (not the FILE NAME). A good list generator will suck this data from the ID3 tag if there is one, and if not it will take the file name with the extension chopped off.

The second line of this pair (the third line) is the actual file name of the media in question. In my example they aren't fully qualified because I run this list by typing "noatun foo.m3u" in my home directory and my music is in ~/mp3, so it just follows the paths as relative from the path of invocation.

ASX

Description: One of the three Windows Media metafile formats. The three are ASX, WAX, and WVX. These formats are identical except for the type of content they may point to. ASX may only point to .asf content.

The ASX family of formats are somewhat a moving target, however, since they are defined by implementation in the windows Media Player and related APIs, which is changing rapidly at this time. I suspect but haven't confirmed that just about any kind of media can be pointed to within a playlist as long as the playlist is correctly handed off from the browser to Windows Media Player.

Example:

```
<ASX version = "3.0">
<TITLE>Simple ASX Demo</TITLE>
    <ENTRY>
        <TITLE>An Entry in a Simple ASX</TITLE>
        <AUTHOR>Microsoft Corporation</AUTHOR>
        <COPYRIGHT>(c)2003 Microsoft Corporation</COPYRIGHT>
        <!-- This is a comment.  Change the following path to point to your ASF -->
        <REF HREF = "mms://windowsmediaserver/path/yourfile.asf" />
    </ENTRY>
</ASX>
```

Mime type: `video/x-ms-asf (also used for .asf files)`

Distinguishing features: Can point to .asf files ONLY.

Definition URL:

1. The canonical reference can be found by searching for "ASX" on Microsoft.com. I have found that giving a specific URL here is pointless, because they go bad quickly.

2. http://cita.rehab.uiuc.edu/mediaplayer/asx-overview.html

Originator: Microsoft

Implementations: Various Microsoft media players

Metadata support: Predefined fields related to entries including author, title, and copyright information.

WAX

Description: Exactly like ASX and WVX, except that it can only contain references to .asf or .wma, but NOT to .wmv files. See ASX reference for more details.

Example:

```
<ASX version = "3.0">
<TITLE>Simple WAX Demo</TITLE>

    <ENTRY>
        <TITLE>An ASF Entry in a Simple WAX</TITLE>
        <AUTHOR>Microsoft Corporation</AUTHOR>
        <COPYRIGHT>(c)2003 Microsoft Corporation</COPYRIGHT>
        <!-- This is a comment.  Change the following path to point to your ASF -->
        <REF HREF = "mms://windowsmediaserver/path/yourfile.asf" />
    </ENTRY>

    <ENTRY>
        <TITLE>A WMA Entry in a Simple WAX</TITLE>
        <AUTHOR>Microsoft Corporation</AUTHOR>
        <COPYRIGHT>(c)2003 Microsoft Corporation</COPYRIGHT>
        <!-- This is a comment.  Change the following path to point to your WMA -->
        <REF HREF = "mms://windowsmediaserver/path/yourfile.wma" />
    </ENTRY>

</ASX>
```

Mime type: `audio/x-ms-wax`

Distinguishing features: Can only contain references to .asf or .wma files.

Definition URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmplay/mmp_sdk/creatingmetafileplaylists.asp

Originator: Microsoft

Implementations: Various Microsoft media players

Metadata support: Predefined fields related to entries including author, title, and copyright information.

WVX

Description: Exactly like ASX and WAX, except that it can contain references to .asf, .wma, and .wmv files. See ASX reference for more details.

Example:

```
<ASX version = "3.0">
<TITLE>Simple WVX Demo</TITLE>

    <ENTRY>
        <TITLE>An ASF Entry in a Simple WVX</TITLE>
        <AUTHOR>Microsoft Corporation</AUTHOR>
        <COPYRIGHT>(c)2003 Microsoft Corporation</COPYRIGHT>
        <!-- This is a comment.  Change the following path to point to your ASF -->
        <REF HREF = "mms://windowsmediaserver/path/yourfile.asf" />
    </ENTRY>

    <ENTRY>
        <TITLE>A WMA Entry in a Simple WVX</TITLE>
        <AUTHOR>Microsoft Corporation</AUTHOR>
        <COPYRIGHT>(c)2003 Microsoft Corporation</COPYRIGHT>
        <!-- This is a comment.  Change the following path to point to your WMA -->
        <REF HREF = "mms://windowsmediaserver/path/yourfile.wma" />
    </ENTRY>

    <ENTRY>
        <TITLE>A WMV Entry in a Simple WVX</TITLE>
        <AUTHOR>Microsoft Corporation</AUTHOR>
        <COPYRIGHT>(c)2003 Microsoft Corporation</COPYRIGHT>
        <!-- This is a comment.  Change the following path to point to your WMV -->
        <REF HREF = "mms://windowsmediaserver/path/yourfile.wmv" />
    </ENTRY>

</ASX>
```

Mime type: `video/x-ms-wvx`

Distinguishing features: Can only contain refs to .asf, .wma, .wmv files.

Definition URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wmplay/mmp_sdk/creatingmetafileplaylists.asp

Originator: Microsoft

Implementations: Various Microsoft media players

Metadata support: Predefined fields related to entries including author, title, and copyright information.

PLS

Description: A proprietary format used for playing Shoutcast and Icecast streams. The syntax of a PLS file is the same syntax as a Windows .ini file and was probably chosen because of support in the Windows API.

Example:
```
[Playlist]
NumberOfEntries=1
File1=http://www.panix.com/web/faq/multimedia/sample.mp3
Title1=Bird Song
Length1=21
Version=2
```

Mime type: audio/x-scpls

Distinguishing features: A Windows .ini file

Definition URL: PLS is documented via reverse engineering. One source of documentation is at http://developer.apple.com/documentation/QuickTime/QT6WhatsNew/Chap1/chapter_1_section_58.html.

Originator: Winamp

Implementations: Winamp, QuickTime

Metadata support: Metadata is included in the entry for each song, in a set of parallel arrays where FileN=[address of file]; TitleN=[title of song].

B4S

Description: A proprietary XML-based format introduced in Winamp version 3.

Example:
```
<?xml version="1.0" encoding='UTF-8' standalone="yes"?>
<WinampXML>
<!-- Generated by: Nullsoft Winamp3 version 3.0c -->
<playlist num_entries="2" label="Playlist 005">

<entry Playstring="file:E:\fresh dls\Modern Rock - March 2003\(modern rock march 2003)-08-system of
a down-i-e-a-i-a-i-o.mp3">
<Name>System Of A Down - I-E-A-I-A-I-O</Name>
<Length>189027</Length>
</entry>
<entry Playstring="file:\CARDS\Albums\normal\Led Zeppelin - Houses Of The Holy\Led Zeppelin - Houses
Of The Holy - 03 - Over The Hills And Far Away.mp3">
<Name>Led Zeppelin - Over The Hills And Far Away</Name>

<Length>289747</Length>
</entry>
</playlist>
</WinampXML>
```

Mime type: Unknown

Distinguishing features: XML. Broken file URIs. Resembles iTunes library XML in many ways.

Definition URL: None. B4S is documented via reverse engineering.

Originator: Winamp

Implementations: Winamp 3+

Metadata support: Predefined fields related to entries including title and song length.

Kapsule

Description: XML manifest used by Kazaa.

Example:

```
<?xml version="1.0"?>
<kapsule version="1.0">
  <metakapsule>
   <publisher>301 Records</publisher>
   <creator>The Honey Palace</creator>
   <title>Have You Seen Love?</title>
   <description>"Psyca-Mojo music", "The Honey Palace"
     debut album, 'Have you seen love?', communicates the dissolving of
     mind and the freeing of heart. There's nothing too heavy or preachy,
     it is simply a great, hypnotic Rock Album!</description>
   <keywords>sunburst freak, freedom child, lovespell,
     warm the one you love, the human locust, world song, me-oh-my, grace,
     9 miles high, quiet friend, 301 records, honey palace</keywords>
   <language>en</language>
   <audio>
     <album>Have You Seen Love?</album>
   </audio>
   <stylesheet>Kapsule.xsl</stylesheet>
   <date>2002-10-01</date>
   <thumbnail>honey_pix1.jpg</thumbnail>
  </metakapsule>
  <item>

   <identifier>urn:topsearch:588d90964313b128571c44f51d59a912641a
     1d1d9e748415a1cba3e56ed64c3875636487</identifier>

   <identifier>http://media6.altnet.com/301/honeypalace/Sunburst
     Freak.wma</identifier>
   <filename>Sunburst Freak.wma</filename>
   <audio>
     <creator>The Honey Palace</creator>
     <album>Have You Seen Love?</album>
     <title>Sunburst Freak</title>
```

```
    <rights>

     <identifier>http://www.thehoneypalace.com/</identifier>
    </rights>
    <fileSize>4397717</fileSize>
   </audio>
  </item>
  <item>

   <identifier>urn:topsearch:8c603512f1f5f0b9f93afc844d17a27021b8
    d257f5c4e95f4269ba0b3b4b25941ba87b90</identifier>

   <identifier>http://media1.altnet.com/301/honeypalace/Freedom
    Child.wma</identifier>
   <filename>Freedom Child.wma</filename>
   <audio>
    <creator>The Honey Palace</creator>
    <album>Have You Seen Love?</album>
    <title>Freedom Child</title>
    <rights>

     <identifier>http://www.thehoneypalace.com/</identifier>
    </rights>
    <fileSize>3357890</fileSize>
   </audio>
  </item>
  <item>

   <identifier>urn:topsearch:80390185925144be2c46b82ca7386252cebf
    7fb36ec670adfcac673bfa8074650d514844</identifier>

   <identifier>http://media3.altnet.com/301/honeypalace/Love
    Spell.wma</identifier>
   <filename>Love Spell.wma</filename>
   <audio>
    <creator>The Honey Palace</creator>
    <album>Have You Seen Love?</album>
    <title>Love Spell</title>
    <rights>

     <identifier>http://www.thehoneypalace.com/</identifier>
    </rights>
    <fileSize>6220385</fileSize>
   </audio>
  </item>

</kapsule>
```

Mime type: None. (This format is intended for use in Kazaa network, not the web).

Distinguishing features: Audio resources are identified either by a URL or by a hash; the identifier element corresponds to the xt argument in a Magnet URI. Intended for use on decentralized networks, so unusually network-friendly and oriented towards hypertext for a playlist format. Modern and clean XML, though it unfortunately does not use namespaces to reuse elements well defined in other places, like rights URL and artist metadata.

Definition URL: http://groups.yahoo.com/group/magnet-uri/message/86

Originator: Kazaa

Implementations: Kazaa

Metadata support: Predefined elements for creator, album, title, rights, filesize.

KPL

Description: Kazaa Playlist Format. Like PLS, it is in Windows .ini format.

Example:

```
[Metadata]
artist=Too Much Joy
album=Live at Least (Outtakes)
category=
language=All
year=
keywords=
description=Tracks from http://www.sayhername.com/tmj_live.php
[Entries]
entry1=Too_Much_Joy_-_Live_at_Least_-_Outtakes_-
Susquehanna_Hat_Company.mp3\382\fa430dab2be8b5f019a40e075911598c4749b\
Susquehanna Hat Company\Too Much Joy\Live at Least - Outtakes\0:03:02
entry2=Too_Much_Joy_-_Live_at_Least_-_Outtakes_-_
Seasons_in_the_Sun.mp3\ab8594bc\f41dc01ba5f5ec7837a324424ae3d06f\
Seasons in the Sun\Too Much Joy\Live at Least - Outtakes\Unknown
entry3=Too_Much_Joy_-_Live_at_Least_-_Outtakes_-_
What_It_Is.mp3\006526823c536f25\6766c994ce1b633db132e16c\What It Is\
Too Much Joy\Live at Least - Outtakes\0:04:28
entry4=Too_Much_Joy_-_Live_at_Least_-_Outtakes_-_
Secret_Handshake.mp3\6832cef43d\f690f877a4483092e171f5ccfac3e7\Secret Handshake\
Too Much Joy\Live at Least - Outtakes\Unknown
```

Mime type: Unknown

Distinguishing features: To enable files to be accessible via P2P networks, uses secure hashes to identify them.

I have seen a report that all KPL files contain the unique hex string 5B 4D 65 74 61 64 61 74 61 5D 0D 0A 61 72 74 69 73 74 3D, but since I don't have a sample of a complete KPL file cannot verify that.

Definition URL: There is no published definition. The information for this entry was drawn from http://groups.yahoo.com/group/magnet-uri/message/27.

Originator: Kazaa

Implementations: Kazaa

Metadata support: Predefined fields including artist, album, category, language, year, keywords, and description. These correspond roughly to ID3v1 tags, which support song title, artist, album, genre, year, and comment.

MAGMA

Description: MAGMA is a manifest format proposed informally by Gordon Mohr on the Magnet-URI mailing list. The purpose is to allow Magnet URI types to be used in playlists. Like KPL, these would allow files to be identified by hash. Like M3U and RAM, these would be one-dimensional. Like the ASX family and RAM, the MAGMA format would be used to flag the need for a special handler.

MAGMA makes two improvements to M3U syntax. First, the header line contains a magic number (the "MAGMA" string) to identify the file type, as well as a URI to identify the web source of the file. Second, an entry can span multiple lines if trailing parts begin with whitespace.

Example:

```
#MAGMA magnet:?mt=.&dn=Lisa%20Rein's%20Downloadable%20MP3s&as=http://xavvy.com/lisarein.magma

# these are all available from http://www.lisarein.com

magnet:?xt=urn:sha1:CWRXLWCZZDOAL7PHJNMWHAOQH6HNJETJ
   &dn=Lisa%20Rein%20%2d%20Shake%20All%20Over%2emp3
   &as=http://www.lisarein.com/LisaRein-ShakeAllOver.mp3

magnet:?xt=urn:sha1:E7QPNLKNYZIXAFMMPFJFJG3P2MEJ7XF2
   &dn=Lisa%20Rein%20%2d%20Number%202%20%28The%20Ballad%20of%20the%20Monsturd%29%2emp3
   &as=http://www.finetuning.com/number2.mp3
```

Mime type: None

Distinguishing features: MAGNET URIs, a magic number in the header line, ability to have line breaks within an entry.

Definition URL: http://groups.yahoo.com/group/magnet-uri/message/31

Originator: Gordon Mohr, in email

Implementations: None

Metadata support: None

RAM

Description: A RAM file is a flat file containing a list of media URLs, with one URL per line. It is almost identical to an M3U playlist, except that it may contain URLs of proprietary RealAudio media types, and URLs can be tweaked to affect the Real player startup mode.

Notice that this difference between M3U and RAM is similar to the way that Microsoft playlist formats like ASX, WMV and WAX have the same syntax but are constrained to point towards different kinds of remote resources.

Startup mode of the Real client can be specified by adding a query string after the resource. RAM embeds parameters for the local player in URLs of remote resources; this practice can be described as bizarre.

RAM is a loosely defined proprietary format whose purpose can be summed up as launching one of the various Real clients and having it figure out what to do.

Example:

```
# This is a real audio streaming resource served by a proprietary real-time protocol
rtsp://ra2.panix.com/tutorial/sample.ra

# This is a static resource
http://www.panix.com/web/faq/multimedia/sample.ra

# This opens a SMIL resource in full-screen mode:
rtsp://realserver.example.com/media/sample1.smil?screensize="full"

# This opens a file on the client.
file:///Users/lgonze/Music/mp3/misc/wtf.mp3
```

Mime type:

```
audio/vnd.rn-realaudio

audio/x-pn-realaudio
```

Distinguishing features: Flat file that can point to RealAudio streams.

Definition URL: http://service.real.com/help/library/guides/production8/htmfiles/server.htm#14171

Originator: Real Audio

Implementations: Real players -- RealPlayer, RealOne, etc

Metadata support: Metadata in RAM files is appended to URIs.

If the URI in the RAM file points to a RealMedia file (i.e. not an MP3, etc.) you can overwrite the title, author, copyright and abstract information in the RealPlayer by adding parameters with these four names (all in lower case) at the end of the line, separated by a ? for the first parameter, and &'s for the rest. For example:

```
pnm://server.address.here:1234/realmedia.rm
 ?title=My Song
 &author=My Artist
 ©right=2004 My Company
 &abstract=Under the (i) button!
```

(This should all be one line, without spaces).

It's not necessary to use all these parameters. Simply adding ?abstract=Info after the filename leaves title, author and copyright unchanged, and only adds "Abstract: Info" in the player's Clip information window.

(This entry on metadata in RAM is from René Davids via email).

SMIL

Description: An XML format with a fair amount of overlap with HTML. Alone among playlist types, SMIL is an open standard, in this case from the W3C. According to the W3C, SMIL "is typically used for rich media/multimedia presentations which integrate streaming audio and video with images, text or any other media type".

Example:

```
<smil>
<body>
<seq>

<audio src="http://example.com/foo.mp3"/>
<audio src="http://example.com/bar.mp3"/>
</seq>
</body>
</smil>
```

Mime type: `application/smil`

Distinguishing features: XML, an open standard, a high degree of power and flexibility.

Definition URL: http://www.w3.org/AudioVideo/

Originator: W3C

Implementations: Various Real media players, QuickTime, Windows Media Framework API, a number of alpha-level open source implementations.

Metadata support: SMIL allows but doesn't define embedded metadata using RDF. Because of this approach, it has by far the most powerful and flexible metadata support of any playlist format.

HTML+Time

Description: A variation of SMIL embeddable in HTML. HTML+Time is nominally an open standard, having been submitted to the W3C in 1998 and published (but not recommended or adopted) by the W3C as a note for discussion.

Example: (from http://msdn.microsoft.com/workshop/author/behaviors/time.asp)

```
<HTML>
<HEAD>
<STYLE>
        .time    {behavior: url(#default#time2);}
</STYLE>
</HEAD>
<BODY>
<P>This text appears right away. More lines to follow...</P>
<P CLASS="time" BEGIN="2" DUR="5" >This appears after 2 seconds.</P>
<P CLASS="time" BEGIN="4" DUR="5">This appears after 4 seconds.</P>
<P CLASS="time" BEGIN="6" DUR="5">This appears after 6 seconds.</P>
<P>This is the last line.</P>
</BODY>
</HTML>
```

Mime type: `text/html`

Distinguishing features: A hybrid of two open standards. Incorporation of playlist semantics into HTML allows in-browser rendering.

Definition URL: http://www.w3.org/TR/NOTE-HTMLplusTIME

Originator: Microsoft

Implementations: Internet Explorer 5.5+

Metadata support:

iTunes library

Description: iTunes library data. XML based. Proprietary.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>Major Version</key><integer>1</integer>
        <key>Minor Version</key><integer>1</integer>
        <key>Application Version</key><string>4.0</string>
        <key>Tracks</key>
        <dict>
                <key>50000</key>
                <dict>
                        <key>Track ID</key><integer>5000</integer>
                        <key>Name</key><string>wtf</string>
                        <key>Artist</key><string>Madonna</string>
                        <key>Kind</key><string>MPEG audio stream</string>
                        <key>Total Time</key><integer>10031</integer>
                        <key>Date Added</key><date>2003-06-22T20:36:34Z</date>
                        <key>Bit Rate</key><integer>128</integer>
                        <key>Sample Rate</key><integer>44100</integer>
                        <key>Play Count</key><integer>5</integer>
                        <key>Play Date</key><integer>-1145441292</integer>
```

```
                        <key>Play Date UTC</key><date>2003-10-21T00:20:04Z</date>
                        <key>Location</key><string>http://gonze.com/blog/music/Madonna-WTF.mp3</string>
                </dict>
                <key>50001</key>
                <dict>
                        <key>Track ID</key><integer>50001</integer>
                        <key>Kind</key><string>MPEG audio stream</string>
                        <key>Location</key><string>http://www.epitonic.com/files/reg/songs/mp3/
Bad_Brains-Pay_To_Cum.mp3</string>
                </dict>
        </dict>
        <key>Playlists</key>
        <array>
                <dict>
                        <key>Name</key><string>very hacked playlist</string>
                        <key>All Items</key><true/>
                        <key>Playlist Items</key>
                        <array>
                                <dict>
                                        <key>Track ID</key><integer>50000</integer>
                                </dict>
                                <dict>
                                        <key>Track ID</key><integer>50001</integer>
                                </dict>
                        </array>
                </dict>
        </array>
</dict>
</plist>
```

Mime type: None. iTunes playlists are not intended to be transferred over a network.

Distinguishing features: A catalog in XML format of song files on the local machine. Permits URLs of remote resources to be stored. Metadata is extracted from song files when the catalog is built and thereafter gotten from the catalog.

Definition URL: The iTunes library format is built on top of Apple's proprietary plist format. Aside from plist documentation, the iTunes library format is documented by reverse engineering.

Originator: Apple

Implementations: iTunes software. Numerous open source implementations.

Metadata support: Song metadata is extracted from audio files and mirrored within the catalog.

DAAP

Description: The DAAP protocol for music sharing used by iTunes and open source clones contains, in addition to other functionality, a binary syntax expressing the same information as in the iTunes library XML format.

Metadata is obtained by requesting the resource "/databases/Database ID/items" from a DAAP server.

Example:

```
daap.databasesongs                      0x00031268
 dmap.status                            0x00000004    number    0x000000c8(200)
 dmap.updatetype                        0x00000001    number    0x00(0)
 dmap.specifiedtotalcount               0x00000004    number    0x000001cf(463)
 dmap.returnedcount                     0x00000004    number    0x000001cf(463)
 dmap.listing                           0x00031233
  dmap.listingitem                      0x00000173
   dmap.itemkind                        0x00000001    number    0x02(2)
   daap.songalbum                       0x0000000c    string    "American Pie"
   daap.songartist                      0x00000010    string    "Allison Hannigan"
   daap.songbeatsperminute              0x00000002    number    0x0000(0)
   daap.songbitrate                     0x00000002    number    0x0080(128)
   daap.songcomment                     0x00000000    string    ""
   daap.songcompilation                 0x00000001    number    0x00(0)
   daap.songcomposer                    0x00000000    string    ""
   daap.songdateadded                   0x00000004    time      Fri Apr 11 00:17:04 2003
   daap.songdatemodified                0x00000004    time      Fri Apr 11 00:16:47 2003
   daap.songdisccount                   0x00000002    number    0x0000(0)
   daap.songdiscnumber                  0x00000002    number    0x0000(0)
   daap.songdatakind                    0x00000001    number    0x00(0)
   daap.songformat                      0x00000003    string    "mp3"

   daap.songeqpreset                    0x00000000    string    ""
   daap.songgenre                       0x0000000c    string    "Movie Quotes"
   dmap.itemid                          0x00000004    number    0x00000021(33)
   daap.songdescription                 0x0000000f    string    "MPEG audio file"
   dmap.itemname                        0x0000001d    string    "This One Time at Band Camp..."
   com.apple.itunes.norm-volume         0x00000004    number    0x00000000(0)
   dmap.persistentid                    0x00000008    number    0xed8a70f9fca15729(-1330126520946960599)
   daap.songdisabled                    0x00000001    number    0x00(0)
   daap.songrelativevolume              0x00000001    number    0x00(0)
   daap.songstoptime                    0x00000004    number    0x00000000(0)
   daap.songtime                        0x00000004    number    0x00004cbc(19644)
   daap.songtrackcount                  0x00000002    number    0x0000(0)
   daap.songtracknumber                 0x00000002    number    0x0000(0)
   daap.songuserrating                  0x00000001    number    0x00(0)
   daap.songyear                        0x00000002    number    0x0000(0)
```

Mime type: None. DAAP playlists are returned as a portion of a larger set of binary octets.

Distinguishing features: Binary, semantics of iTunes library

Definition URL: http://molelog.molehill.org/blox/Computers/Macintosh/DAAP3.writeback

Originator: Apple

Implementations: iTunes

Metadata support: Embedded in playlist along with resource ID listings.

Creative Commons RDF

Description: The Creative Commons licensing project includes a template for RDF metadata to be embedded in XML or XML-like environments. It happens that when the syntax and semantics of this template are applied to an collection of audio works, the result is indistinguishable from a playlist. As an example, Mike Linksvayer of Creative Commons used an XSLT stylesheet to create a SMIL playlist from the embedded license RDF in the web page for the Creative Commons CD release "Copy Me Remix Me" at http://creativecommons.org/extras/copyremix. This work is included here because it is provocative and original.

Example:

```
<rdf:RDF xmlns="http://web.resource.org/cc/"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

   <Work rdf:about="urn:sha1:BJ5VVVC26NXKOKYGCODPU6JDP4ZLHABM">
      <dc:date>2002</dc:date>
      <dc:format>audio/mp3</dc:format>
      <dc:identifier>http://mirrors.creativecommons.org/copyremix/Analogue_Popkick_-
_Two_June.mp3</dc:identifier>
      <dc:rights><Agent><dc:title>Analogue Popkick</dc:title></Agent></dc:rights>
      <dc:title>Two June</dc:title>
      <dc:type rdf:resource="http://purl.org/dc/dcmitype/Sound" />
      <license rdf:resource="http://creativecommons.org/licenses/by-nc-sa/1.0/" />
   </Work>

   <Work rdf:about="urn:sha1:3LX4N6D4LOEIDYEXSXH7V2Y22NWFMMUM">
      <dc:date>2002</dc:date>
      <dc:format>audio/mp3</dc:format>
      <dc:identifier>http://mirrors.creativecommons.org/copyremix/
Bm_RELOCATION_PROGRAM_-_Superego_Exchange.mp3</dc:identifier>
      <dc:rights><Agent><dc:title>Bm RELOCATION PROGRAM</dc:title></Agent></dc:rights>
      <dc:title>superego exchange</dc:title>
      <dc:type rdf:resource="http://purl.org/dc/dcmitype/Sound" />
      <license rdf:resource="http://creativecommons.org/licenses/by-sa/1.0/" />
   </Work>

   <Work rdf:about="urn:sha1:GQWLXQ4XP74QKZIBOUXF67GKX2SCYZVY">
      <dc:date>2002</dc:date>
      <dc:format>audio/mp3</dc:format>
      <dc:identifier>http://mirrors.creativecommons.org/copyremix/
Clyde_Federal_-_Staten_Island_Ferry.mp3</dc:identifier>
      <dc:rights><Agent><dc:title>Clyde Federal</dc:title></Agent></dc:rights>
      <dc:title>Staten Island Ferry</dc:title>
      <dc:type rdf:resource="http://purl.org/dc/dcmitype/Sound" />
      <license rdf:resource="http://creativecommons.org/licenses/by-nc-sa/1.0/" />
   </Work>

   <License rdf:about="http://creativecommons.org/licenses/by-nc-sa/1.0/">
      <permits rdf:resource="http://web.resource.org/cc/Reproduction" />
      <permits rdf:resource="http://web.resource.org/cc/Distribution" />
      <requires rdf:resource="http://web.resource.org/cc/Notice" />
      <requires rdf:resource="http://web.resource.org/cc/Attribution" />
      <prohibits rdf:resource="http://web.resource.org/cc/CommercialUse" />
      <permits rdf:resource="http://web.resource.org/cc/DerivativeWorks" />
      <requires rdf:resource="http://web.resource.org/cc/ShareAlike" />
   </License>
   <License rdf:about="http://creativecommons.org/licenses/by-sa/1.0/">
      <permits rdf:resource="http://web.resource.org/cc/Reproduction" />
      <permits rdf:resource="http://web.resource.org/cc/Distribution" />
      <requires rdf:resource="http://web.resource.org/cc/Notice" />
      <requires rdf:resource="http://web.resource.org/cc/Attribution" />
      <permits rdf:resource="http://web.resource.org/cc/DerivativeWorks" />
      <requires rdf:resource="http://web.resource.org/cc/ShareAlike" />
   </License>
</rdf:RDF>
```

Mime type: application/rdf+xml.

Distinguishing features: RDF. Makes the metadata the primary object and leaves playlist functionality as one potential application among many.

Definition URL: http://creativecommons.org/technology/metadata/schema.rdf

Originator: Creative Commons

Implementations: A proof of concept stylesheet cached at http://gonze.com/how_he_did_it.html.

Metadata support: All Dublin Core and Creative Commons elements may be applied to the playlist as a whole or to referenced elements.

MusicBrainz metadata

Description: MusicBrainz is a project to compile high quality audio metadata. The MusicBrainz metadata format is not strictly a playlist format at all, but rather a format to describe audio resources which happens to encompass collections of audio resources as well. In that context the collections that it describes are termed *albums*, where *album* is a shorthand for a set of audio resources released together on hard media like CDs, records, or cassettes.

Example: (From http://www.musicbrainz.org/MM/mq_examples.html#track)

```
<mm:Album rdf:about="http://musicbrainz.org/album/e962354a-28f2-44b1-9a26-c8092de4d4f3">
  <dc:title>Sour Times (Nobody Loves Me)</dc:title>
  <dc:creator
   rdf:resource="http://musicbrainz.org/artist/8f6bd1e4-fbe1-4f50-aa9b-94c450ec0f11"/>
  <mm:trackList>
    <rdf:Seq>
      <rdf:li rdf:resource="http://musicbrainz.org/mm-2.1/track/230365bb-caf3-41a9-9c27-51294f9954e1"/>
      <rdf:li rdf:resource="http://musicbrainz.org/mm-2.1/track/d45e3f81-5f21-4eaf-bd07-283ce4b26b02"/>
      <rdf:li rdf:resource="http://musicbrainz.org/mm-2.1/track/5eb94502-c632-4751-ab50-67747171c6a7"/>
      <rdf:li rdf:resource="http://musicbrainz.org/mm-2.1/track/c22502a8-babf-4a96-bd40-f7e879cfa865"/>
      <rdf:li rdf:resource="http://musicbrainz.org/mm-2.1/track/33f21b05-82fa-4f63-bedb-e093a646185b"/>
      <rdf:li rdf:resource="http://musicbrainz.org/mm-2.1/track/c233597a-1004-40dd-985d-2a5eb1b8da48"/>
    </rdf:Seq>
  </mm:trackList>
</mm:Album>
```

Mime type: `application/rdf+xml`

Distinguishing features:

- Use of standards like Dublin Core and RDF.
- Normalized data structure with definitions of tracks, artists, and collections kept separate.
- RDF.
- Hypertext oriented; use of URIs to name objects rather than free text.
- Adheres to best practices for open data formats.

Definition URL: http://www.musicbrainz.org/MM/

Originator: MusicBrainz

Implementations: MusicBrainz

Metadata support: All Dublin Core- and RDF- defined fields. Defines a new namespace for Artist, Album, Track, TrackListing and TrackNum.