-OSTA.org-
Optical Storage
Technology Association

MultiAudio Specification

Revision 1.10

September 16, 2002

POINTS OF CONTACT

OSTA	***Technical Content***
David Bunzel OSTA President 311 East Carrillo Street Santa Barbara, CA 93101 Tel: +1 805 963 3853 Fax: +1 805 962 1541 Email: ray@osta.org http://www.osta.org	Felix Nemirovsky Chairman, MultiRead Subcommittee Oak Technology, Inc. 139 Kifer Court Sunnyvale, CA 94086 Tel: +1 415 643 0944 Fax: +1 520 299 4319 Email: felixn@oaktech.com

ABSTRACT

This specification defines a format for organizing compressed audio on an optical disc. The applicable clauses of the specification containing the word "*shall*" are the requirements to be compliant with the format.

LICENSING

IMPORTANT NOTICES

Contents

List of Figures

1 General

1.1 Scope

This document specifies the *MultiAudio* (*MA*) format for organizing compressed audio on an optical media.

Note: *The medium is not restricted to be in only one type, i.e. the medium may be write–once or rewritable.*

The *MA* format is not a filesystem replacement. *MA* structures are being stored as files in a filesystem supported by a medium. The format is filesystem independent, i.e. there are no explicit references between *MA* structures and a filesystem they are stored on.

Note: *If the medium supports more than one filesystem then each filesystem that is meant to provide MA compliance shall contain all MA files within its namespace.*

1.2 Purpose

The intention of this specification is to facilitate the use of removable optical media with compressed audio files by consumer devices such as CD and DVD players. Such discs could be initialized and played the same way on PC (including non-Windows OS) and CE devices. The purpose of this format is to create a standard disc structure for all authoring software and playback devices to facilitate uniformity of play back. Thus, such a disc may contain hundreds of songs but would still be as easy to play as a Red Book Audio disc or to navigate as a VideoCD or DVD Video disc.

1.3 References

ISO 646 – Information processing – ISO 7–bit coded character set for information interchange.

ISO 10646 – Information technology – Universal Multiple–Octet Coded Character Set (UCS).

Unicode – The Unicode Standard, Version 3.0 (ISBN 0-201-61633-5).

1.4 Definitions

1.4.1 0–Based Number

The first element is numbered 0, the second is numbered 1, etc.

1.4.2 1–Based Number

The first element is numbered 1, the second is numbered 2, etc.

1.4.3 Basic Data Types

Type	Description
`byte`	A 8–bit value.
`Uint8`	An unsigned 8–bit integer.
`Sint8`	A signed 8–bit integer.
`Uint16`	An unsigned 16–bit (two byte) integer. The value shall be recorded in little endian (LSB first) format.
`Sint16`	A signed 16–bit (two byte) integer. The value shall be recorded in little endian (LSB first) format.
`Uint32`	An unsigned 32–bit (four byte) integer. The value shall be recorded in little endian (LSB first) format.
`Sint32`	A signed 32–bit (four byte) integer. The value shall be recorded in little endian (LSB first) format.

1.4.4 TOC

MultiAudio Table of Contents. Data structure written on a Compact Disc to organize compressed audio files and play lists. Used for retrieval, management and playback of compressed audio files and play lists by CD/DVD consumer player.

1.4.5 Chunk

A tagged structure. A tag allows for identifying a type of the structure and its ordinal number (Section 2.3).

1.4.6 Compressed Audio Disc

Disc containing compressed audio files such as MP3, WMA or other non-PCM audio files.

1.4.7 Extent

A continuous collection of sectors on the disc.

1.4.8 Extra Data

TOC structures allow for accommodating extensions which may be placed in an *Extra Data* area forming a sequence of extensions. Each extension shall be a *Chunk* (see Section 2.3) and have its *Ordinal Number* set to its number in the sequence of extensions.

1.4.9 Genre

A description of a classification of a song. For example, classical, jazz, etc.

1.4.10 int

An expression $int(S)$ means truncating S to the integer part.

1.4.11 Number

A number in this document (unless explicitly stated):

- is in a decimal format if written without any suffixes,
- is in a hexadecimal format if written with a suffix 'h',
- is in a binary format if written with a suffix 'b'.

1.4.12 Playlist

A named collection of references to *Tracks* (Section 2.6).

1.4.13 rem

An expression $rem(S, D)$ means the integer reminder of dividing S by D.

1.4.14 String

A string is one of

- A sequence of *d–characters* defined by ISO–646 as characters 'A' through 'Z' and the underscore '_' character, terminated by a null character (00h). *d–characters* are used to define encoding type of a *TrackEntry*.
- A sequence of 7–bit ASCII characters referred to further as *ASCII*, terminated by a null character (00h).

- A sequence of UTF–16 encoded characters referred to as *UNICODE*, terminated by a wide null character (a sequence of two 00h bytes). The characters may be recorded in either little endian or big endian formats (the *byte order mark* – an initial byte sequence FEFFh – is used to distinguish between the two byte orders).

A string length in this document is length in bytes including terminating character.

An empty string shall be represented by the terminating character.

Note: *The reason for terminating null(s) is that there are a lot of libraries handling null–terminated strings.*

1.4.15 Track

A collection of audio data.

1.4.16 Tracklist

A named collection of track entries (structures described in Section 2.5). This structure contains the same fields as *Playlist* except for full *TrackEntries* that replace track indexes.

1.4.17 Tracklist Directory

A structure describing a collection of *Tracklists* (Section 2.8).

1.5 Terms

Unicode – a coded character set specified by a consortium of major computer corporations, software producers, database vendors, etc. From version 1.1 on, Unicode is kept compatible with ISO/IEC 10646 and its extensions.

UCS–2 – ISO/IEC 10646 encoding form: Universal Character Set coded in 2 octets.

UTF–16 – Unicode (or UCS) Transformation Format, 16-bit encoding form. The UTF-16 is the Unicode Transformation Format that serializes a Unicode scalar value (code point) as a sequence of two bytes, in either big-endian or little-endian format.

2 MultiAudio Structures

2.1 DateAndTime format

#	RBP	Length	Name	Contents
1	0	2	Type And Timezone	Uint16
2	2	2	Year	Uint16
3	4	1	Month	Uint8
4	5	1	Day	Uint8
5	6	1	Hour	Uint8
6	7	1	Minute	Uint8
7	8	1	Second	Uint8
8	9	1	Centiseconds	Uint8
9	10	1	Hundreths Of Microseconds	Uint8
10	11	1	Microseconds	Uint8

2.1.1 Type And Timezone

Ref: ISO 13346 1/7.3.

Type refers to the most significant 4 bits of this field, and *TimeZone* refers to the least significant 12 bits of this field.

The *Type* shall be set to 1. The value of 1 indicates that the structure shall be interpreted as local time.

The *TimeZone* shall be interpreted as a signed value and shall be set to -2047 if no timezone is specified or to 1–minute increments from Coordinated Universal Time.

2.1.2 Year

Numeric representation of a year.

2.1.3 Month

Numeric representation of a month.

2.1.4 Day

Numeric representation of a day of the month.

2.1.5 Hour

Numeric representation of a hour.

2.1.6 Minute

Numeric representation of a minute.

2.1.7 Second

Numeric representation of a second.

2.1.8 Centiseconds

Numeric representation of a centiseconds.

2.1.9 Hundreths Of Microseconds

Numeric representation of hundreths of a second.

2.1.10 Microseconds

Numeric representation of microseconds.

2.2 Tag

#	RBP	Length	Name	Contents
1	0	4	Identifier	`Uint32`
2	4	2	Ordinal Number	`Uint16`
3	6	2	Reserved	`Uint16`
4	8	4	Length	`Uint32`

2.2.1 Identifier

This field shall contain a 32 bit identifier. Identifiers in [00000000h – 7FFFFFFFh] range are reserved for OSTA use. In the reserved range, values [0000h – 01000000] are used to represent *MA* types, values [01000000h – 7FFFFFFFh] are used to signature *MA* files.

Known *MA* structure identifiers are:

Identifier	Description
5F544F43h	*TOC_Header* (Section 2.4)
2	*TrackEntry* (Section 2.5)
3	*Playlist* (Section 2.6)
4	*TracklistDirectory* (Section 2.6)
54524C53h	*Tracklist* (Section 2.7)
5	*ExtraData* (Section 2.9)

2.2.2 Ordinal Number

This field may be used to determine an ordinal number of a structure with this tag in a sequence.

2.2.3 Reserved

This is a reserved field and shall be set to 0.

2.2.4 Length

This field shall contain the length of a structure containing the tag.

2.3 Chunk

A *Chunk* is a generic data type utilized by this document. Each *MA* structure is a *Chunk*. Each future data extension shall also be a *Chunk*.

The length of each *Chunk* shall be evenly divisible by four.

It is possible for users/applications to define "private" extensions. Each of such a private extension shall be a *Chunk* too and have tag identifier from the [10000000h – FFFFFFFFh] range. Private extensions shall be located in *Extra Data* areas (Section 2.9).

#	RBP	Length	Name	Contents
1	0	12	Tag	`tag`
2	12	*L_D*	Data	`byte`

2.3.1 Tag

`Tag.Identifier` subfield shall be set to a proper value for the structure (see table in Section 2.2.1)..

`Tag.OrdinalNumber` subfield shall be set to a 0–based ordinal number of that structure within its sequence.

`Tag.Length` subfield shall be set to $8 + L\_D$ structure.

2.3.2 Data

Structure data. The length L_D shall be evenly divisible by four.

2.4 TOC_Header format

#	RBP	Length	Name	Contents
1	0	12	Tag	`tag`
2	12	2	Version Number	`Uint16`
3	14	36	UUID	`byte`
4	50	4	Length of TOC	`Uint32`
5	54	2	Text Format	`Uint16`
6	56	128	Volume Name	`byte`
7	184	128	Data Preparer Identifier	`byte`
8	312	128	Publisher Identifier	`byte`
9	440	128	Copyright	`byte`
10	568	12	Creation Date And Time	`DateAndTime`
11	580	12	Modification Date And Time	`DateAndTime`
12	592	12	Effective Date And Time	`DateAndTime`
13	604	12	Expiration Date And Time Expires	`DateAndTime`
14	616	2	Number of Playlist Directories $(=N\_D)$	`Uint16`
15	618	2	Number of Tracks $(=N\_T)$	`Uint16`
16	620	2	Number of Playlists $(=N\_P)$	`Uint16`
17	622	2	Reserved	`Uint16`
18	624	4	Offset to Extra Data	`Uint32`
19	628	4	Flags	`Uint32`
20	632	$N\_D * 4$	Playlist Directory Offsets	`Uint32`
21	$632 + N\_D * 4$	$N\_T * 4$	Track Offsets	`Uint32`
22	$632 + N\_D * 4 + N\_T * 4$	$N\_P * 4$	Playlist Offsets	`Uint32`
23	$632 + N\_D * 4 + N\_T * 4 + N\_P * 4$	L_Ex	Extra Data	`ExtraData`

Note: *Providing information in all the fields is mandatory, unless stated otherwise in a field definition.*

2.4.1 Tag

`Tag.Identifier` subfield shall be set to 5F544F43h.

`Tag.OrdinalNumber` subfield shall be set to 0.

`Tag.Length` subfield shall be set to the length of *TOC_Header* structure.

2.4.2 Version Number

This field shall contain the *MA* standard version number. The version number is represented by a 16 bit unsigned integer of increasing value. The version defined by this document is version 1.10, which would be represented in the structure by the value 110. Subsequently, version 2.00 would be 200 and version 2.01 would be 201.

2.4.3 UUID

A universally unique identier (UUID) format was defined in the Open Software Foundation's Distributed Computing Environment RPC standard also available as ISO-11578, which defines UUIDs in an appendix.

This field contains a UUID in 36 byte string format according to that specification (see also Appendix B). When present, the UUID string is not null (00h) terminated.

Providing information in this field is optional. If no string is specified then all bytes of this field shall be set to 0.

2.4.4 Length of TOC

Length of the entire *MA* Table of Contents structure in bytes. This is equivalent to the length of the *TOC.MAU* file in bytes if the file only contains the MultiAudio TOC structure.

2.4.5 Text Format

If this field is set to 0 then volume name shall be interpreted as *ASCII* strings.

If this field is set to 1 then all volume name shall be interpreted as *UNICODE* strings.

2.4.6 Volume Name

This field may contain a user specified disc/volume name. The string shall be encoded according to *Text Format* (2.4.5).

Providing information in this field is optional. If no string is specified then all bytes of this field shall be set to 0.

2.4.7 Data Preparer Identifier

Identifies the person , entity, or application which controlled the preparation of the data, typically the application software or consumer electronics device. The string shall be encoded according to *Text Format* (2.4.5).

Providing information in this field is optional. If no string is specified then all bytes of this field shall be set to 0.

2.4.8 Publisher Identifier

Publisher of the file, often identifies the user who specified what is recorded on the disc. The string shall be encoded according to *Text Format* (2.4.5).

Providing information in this field is optional. If no string is specified then all bytes of this field shall be set to 0.

2.4.9 Copyright

Copright information. The string shall be encoded according to *Text Format* (2.4.5).

Providing information in this field is optional. If no string is specified then all bytes of this field shall be set to 0.

2.4.10 Creation Date And Time

Date and time of the day at which the information in the volume was created.

Providing data in this structure is optional. If no date is specified all the fields shall be set to 0.

2.4.11 Modification Date And Time

Date and time of the day at which the information in the volume was modified.

Providing data in this structure is optional. If no date is specified all the fields shall be set to 0.

2.4.12 Effective Date And Time

Date and time of the day at which the information in the volume may be used.

Providing data in this structure is optional. If no date is specified all the fields shall be set to 0.

2.4.13 Expiration Date And Time Expires

Date and time of the day at which the information in the volume may be regarded as obsolete.

Providing data in this structure is optional. If no date is specified all the fields shall be set to 0.

2.4.14 Number of Playlist Directories

This field shall be set to the number of playlist directories (Section 2.8).

2.4.15 Number of Tracks

This field shall be set to the number of tracks (Section 2.5) on the disc.

2.4.16 Number of Playlists

This field shall be set to the number of playlists (Section 2.6) on the disc, including the default playlist.

2.4.17 Reserved

This field is reserved.

2.4.18 Offset to Extra Data

This field shall be set to the offset (in bytes, from the beginning of *TOC_Header*) to the *Extra Data* field.

2.4.19 Flags

The bit meaning is described in the table below.

0–31	Reserved

Reserved bits shall be set to 0.

2.4.20 Playlist Directory Offsets

This field shall contain a list of playlist directory offsets within the *TOC* structure *from the byte 0* of the *TOC* structure.

Note, the byte 0 of the *TOC_Header* is also the byte 0 of *TOC* structure.

2.4.21 Track Offsets

This field shall contain a list of track byte offsets within the *TOC* structure *from the byte 0* of the *TOC* structure.

Note, the byte 0 of the *TOC_Header* is also the byte 0 of *TOC* structure.

2.4.22 Playlist Offsets

This field shall contain a list of playlist byte offsets within the *TOC* structure *from the byte 0* of the *TOC* structure.

Note, the byte 0 of the *TOC_Header* is also the byte 0 of *TOC* structure.

2.4.23 Extra Data

A *Chunk* containing an array of *Chunk* based structures (Section 2.9).

2.5 TrackEntry format

#	RBP	Length	Name	Contents
1	0	12	Tag	`tag`
2	12	2	Reserved	`Uint16`
3	14	2	Number of Channels	`Uint16`
4	16	4	Average Encoded Bitrate	`Uint32`
5	20	4	Maximum Bitrate	`Uint32`
6	24	4	Sample Rate	`Uint32`
7	28	4	Playing Time	`Uint32`
8	32	2	Text Format	`Uint16`
9	34	2	Offset to Pathname CSD	`Uint16`
10	36	2	Offset to Encoding TID	`Uint16`
11	38	2	Offset to Encoding TID Padding	`Uint16`

12	40	2	Offset to Track Name	`Uint16`
13	42	2	Offset to Performer Name	`Uint16`
14	44	2	Offset to Composer Name	`Uint16`
15	46	2	Offset to Songwriter Name	`Uint16`
16	48	2	Offset to Arranger Name	`Uint16`
17	50	2	Offset to Album Name	`Uint16`
18	52	2	Offset to Genre	`Uint16`
19	54	2	Offset to Pathname	`Uint16`
20	56	2	Offset to the Pathname Padding	`Uint16`
21	58	2	Offset to Extra Data	`Uint16`
22	60	2	Year Recorded	`Uint16`
23	62	2	Track Order	`Uint16`
24	64	*L_Et*	Encoding TID	`byte`
25	64 + *L_Et*	*L_Pet*	Encoding TID Padding	`byte`
26	64 + *L_Et* + *L_Pet*	*L_T*	Track Name	`byte`
27	64 + *L_Et* + *L_Pet* + *L_T*	*L_Pe*	Performer Name	`byte`
28	64 + *L_Et* + *L_Pet*+*L_T*+ *L_Pe*	*L_Co*	Composer Name	`byte`
29	64 + *L_Et* + *L_Pet*+*L_T*+ *L_Pe* + *L_Co*	*L_So*	Songwriter Name	`byte`
30	64 + *L_Et* + *L_Pet*+*L_T*+ *L_Pe*+*L_Co*+ *L_So*	*L_Ar*	Arranger Name	`byte`
31	64 + *L_Et* + *L_Pet*+*L_T*+ *L_Pe*+*L_Co*+ *L_So* + *L_Ar*	*L_Al*	Album Name	`byte`
32	64 + *L_Et* + *L_Pet*+*L_T*+ *L_Pe*+*L_Co*+ *L_So*+*L_Ar*+ *L_Al*	*L_G*	Genre	`byte`
33	64 + *L_Et* + *L_Pet*+*L_T*+ *L_Pe*+*L_Co*+ *L_So*+*L_Ar*+ *L_Al* + *L_G*	*L_lN*	Pathname	`byte`

34	$64 + L\_Et + L\_Pet + L\_T + L\_Pe + L\_Co + L\_So + L\_Ar + L\_Al + L\_G + L\_lN$	*L_Ppn*	Pathname Padding	`byte`
35	$64 + L\_Et + L\_Pet + L\_T + L\_Pe + L\_Co + L\_So + L\_Ar + L\_Al + L\_G + L\_lN + L\_Ppn$	*L_Cs*	Pathname CSD	`byte`
36	$64 + L\_Et + L\_Pet + L\_T + L\_Pe + L\_Co + L\_So + L\_Ar + L\_Al + L\_G + L\_lN + L\_Ppn + L\_Cs$	*L_Ex*	Extra Data	`ExtraData`

Note: *Providing information in all the fields is mandatory, unless stated otherwise in a field definition.*

2.5.1 Tag

`Tag.Identifier` subfield shall be set to 2.

`Tag.OrdinalNumber` subfield shall be set to the 0–based number of the track in the list of tracks.

`Tag.Length` subfield shall be set to the length of this *TrackEntry* structure.

2.5.2 Reserved

Reserved field.

2.5.3 Number of Channels

This field shall contain the number of audio channels.

2.5.4 Average Encoded Bitrate

This field shall contain the average encoded bitrate in bits per second.

2.5.5 Maximum Bitrate

This field shall contain the maximum bitrate in bits per second.

2.5.6 Sample Rate

This field shall contain the sample rate of the track.

2.5.7 Playing Time

This field shall contain the playing time in milliseconds.

2.5.8 Text Format

If this field is set to 0 then text fields except for *Pathname* shall be interpreted as *ASCII* strings.

If this field is set to 1 then all text fields except for *Pathname* shall be interpreted as *UNICODE* strings.

2.5.9 Offset to Pathname CSD

This field is provided for use by filesystems that might use more than one character set for describing pathnames. CSD stands for Character Set Descriptor. Starting from the version 1.10 this is a required field. See Section 4.1 for more information.

2.5.10 Offset to Encoding TID

This field shall be set to the offset (in bytes, from the beginning of *TrackEntry*) to the encoding type identifier. TID stands for Type Identifier.

2.5.11 Offset to Encoding TID Padding

This field shall be set to the offset (in bytes, from the beginning of *TrackEntry*) to the encoding type identifier padding. TID stands for Type Identifier. If the number of padding bytes is zero (padding not present) then the offset to padding shall be set to zero.

2.5.12 Offset to Track Name

This field shall be set to the offset (in bytes, from the beginning of *TrackEntry*) to the *Track Name* string.

2.5.13 Offset to Performer Name

This field shall be set to the offset (in bytes, from the beginning of *TrackEntry*) to the *Performer Name* string.

2.5.14 Offset to Composer Name

This field shall be set to the offset (in bytes, from the beginning of *TrackEntry*) to the *Composer Name* string. *Composer Name* is an optional field. If *Composer Name* is not provided then this field (i.e. offset to *Composer Name*) shall be set to zero.

2.5.15 Offset to Songwriter Name

This field shall be set to the offset (in bytes, from the beginning of *TrackEntry*) to the *Songwriter Name* string. *Songwriter Name* is an optional field. If *Songwriter Name* is not provided then this field (i.e. offset to *Songwriter Name*) shall be set to zero.

2.5.16 Offset to Arranger Name

This field shall be set to the offset (in bytes, from the beginning of *TrackEntry*) to the *Arranger Name* string. *Arranger Name* is an optional field. If *Arranger Name* is not provided then this field (i.e. offset to *Arranger Name*) shall be set to zero.

2.5.17 Offset to Album Name

This field shall be set to the offset (in bytes, from the beginning of *TrackEntry*) to the *Album Name* string.

2.5.18 Offset to Genre

This field shall be set to the offset (in bytes, from the beginning of *TrackEntry*) to the *Genre* in bytes.

2.5.19 Offset to Pathname

This field shall be set to the offset (in bytes, from the beginning of *TrackEntry*) to the *Pathname* string.

2.5.20 Offset to the Pathname Padding

Offset (in bytes, from the beginning of *TrackEntry*) to the padding field. If the number of padding bytes is zero (padding not present) then the offset to padding shall be set to zero.

2.5.21 Offset to Extra Data

This field shall be set to the offset (in bytes, from the beginning of *TrackEntry*) to the *Extra Data* field. If the *Extra Data* field is not provided then this field shall be set to zero.

2.5.22 Year Recorded

This field shall contain a full numeric representation of the year the recording was published. For example 1981 (07BDh). If the year of recording is not available this field shall be set to 0.

2.5.23 Track Order

This field shall contain a numeric representation of the track number from the original album.

2.5.24 Encoding TID

This field shall be encoded as *d–characters*. The contents of that string shall be one of a list of "well known" identifiers (see Appendix A) or a "private" identifier prefixed with `X-`. TID stands for Type Identifier.

2.5.25 Encoding TID Padding

This field provides a 2–byte alignment of the *Track Name*. Since the encoding type identifier is always a string of *d–characters*, it could cause a misalignment of the *Track Name* in section 2.5.26 stored in UNICODE (*Text Format* = 1). For Track Entries with the *Text Format* set to ASCII (*Text Format* = 0), the length of this field is always zero. Computing the length of the padding field is described by the formula:

$$TextFormat * rem(L\_Et, 2)$$

In other words, if the *Text Format* is 0, then the length of the padding is zero. If *Text Format* is 1 (UNICODE strings), then the length of the padding field is 1 if the length of *L_Et* (Length of Encoding TID (Section 2.5.24)) is odd and 0 if *L_Et* is even. The padding value should always be set to 0.

2.5.26 Track Name

This field shall contain the track name according to the character set defined by *Text Format* field.

2.5.27 Performer Name

This field shall contain the *Performer Name* according to the character set defined by *Text Format* field.

2.5.28 Composer Name

This field shall contain the *Composer Name* according to the character set defined by *Text Format* field. This field is optional, i.e. may be either an empty string or may be not present at all (in which case the offset to this field shall be set to zero).

2.5.29 Songwriter Name

This field shall contain the *Song Writer* according to the character set defined by *Text Format* field. This field is optional, i.e. may be either an empty string or may be not present at all (in which case the offset to this field shall be set to zero).

2.5.30 Arranger Name

This field shall contain the *Arranger Name* according to the character set defined by *Text Format* field. This field is optional, i.e. may be either an empty string or may be not present at all (in which case the offset to this field shall be set to zero).

2.5.31 Album Name

This field shall contain the album name according to the character set defined by *Text Format* field.

2.5.32 Genre

This field shall contain the genre description according to the character set defined by *Text Format* field.

2.5.33 Pathname

This field shall contain the pathname encoded in a format supported by the filesystem that contains these *MA* structures.

This field shall contain the relative pathname to the track entry file. The pathname is relative to the *MA* `TOC.MAU` file. Depending on the value of *Pathname CSD* the format of the pathname field is defined by the *Text Format* field (Section 2.5.8) **or** it is stored in the "native" format. Detailed explanation of how to interpret *Pathname CSD* can be found in Section 4.1

2.5.34 Pathname Padding

This field shall contain padding bytes to ensure that the *Extra Data* field starts at 4–byte alignment boundary. The length of that field is

$$int((TempLen + 3)/4) * 4 - TempLen$$

where

$$TempLen = 64 + L\_Et + L\_Pet + L\_T + L\_Pe + L\_Co + L\_So + L\_Ar + L\_Al + L\_G + L\_lN$$

2.5.35 Pathname CSD

This field shall be compatible with the *Chunk* (The Pathname Character Set Descriptor (CSD) is a required field that is compatible with the Chunk structure (Section 2.3). The CSD is used to interpret the format of the Pathname field (Section 2.8.13. The definition of the contents of this structure can be found in Section 4.1.

2.5.36 Extra Data

A *Chunk* containing an array of *Chunk* based structures (Section 2.9).

2.6 Playlist format

The *Playlist* structure contains playlist name, description and a list of tracks. Tracks in the list should be played in the consecutive order.

#	RBP	Length	Name	Contents
1	0	12	Tag	`tag`
2	12	2	Number of Tracks ($=N\_T$)	`Uint16`
3	14	2	Text Format	`Uint16`
4	16	2	Offset to Name	`Uint16`
5	18	2	Offset to Description	`Uint16`

6	20	2	Offset to Padding	`Uint16`
7	22	2	Offset to Track Indexes	`Uint16`
8	24	4	Offset to List Padding	`Uint32`
9	28	4	Offset to Extra Data	`Uint32`
10	32	L_N	Playlist Name	`byte`
11	$32 + L\_N$	L_D	Playlist Description	`byte`
12	$32 + L\_N + L\_D$	L_Pp	Padding	`byte`
13	$32 + L\_N + L\_D + L\_Pp$	$2 * N\_T$	Track Indexes	`Uint16`
14	$32 + L\_N + L\_D + L\_Pp + 2 * N\_T$	L_Pp	List Padding	`byte`
15	$32 + L\_N + L\_D + L\_Pp + 2*N\_T+L\_Pp$	L_Ex	Extra Data	`ExtraData`

Note: *Providing information in all the fields is mandatory, unless stated otherwise in a field definition.*

2.6.1 Tag

`Tag.Identifier` subfield shall be set to 3.

`Tag.OrdinalNumber` subfield shall be set to the 0–based number of the playlist in the list of playlists.

`Tag.Length` subfield shall be set to the length of this *Playlist* structure.

2.6.2 Number of Tracks

This field shall be set to the number of tracks in the play list.

2.6.3 Text Format

If this field is set to 0 then *Play List Name* and *Play List Description* shall be interpreted as *ASCII* strings.

If this field is set to 1 then all *Play List Name* and *Play List Description* shall be interpreted as *UNICODE* strings.

2.6.4 Offset to Name

This field shall be set to the offset (in bytes, from the beginning of *Playlist*) to the *Playlist Name* string.

2.6.5 Offset to Description

This field shall be set to the offset (in bytes, from the beginning of *Playlist*) to the *Play List Description* string.

2.6.6 Offset to Padding

This field shall be set to the offset (in bytes, from the beginning of *Playlist*) to the padding before the array of track indexes. If the number of padding bytes is zero (padding not present) then the offset to padding shall be set to zero.

2.6.7 Offset to Track Indexes

This field shall be set to the offset (in bytes, from the beginning of *Playlist*) to track entry index array.

2.6.8 Offset to List Padding

This field shall be set to the offset (in bytes, from the beginning of *Playlist*) to the padding before *Extra Data* field. If the number of padding bytes is zero (padding not present) then the offset to padding shall be set to zero.

2.6.9 Offset to Extra Data

This field shall be set to the offset (in bytes, from the beginning of *Playlist*) to the *Extra Data* field. If the *Extra Data* field is not provided then this field shall be set to zero.

2.6.10 Playlist Name

This field shall contain the playlist name according to the character set defined by *Text Format* field.

2.6.11 Playlist Description

This field shall contain the play list description according to the character set defined by *Text Format* field.

2.6.12 Padding

This field shall contain padding bytes to ensure that the array of track indexes starts at 4–byte alignment boundary. The length of that field is

$$int((TempLen + 3)/4) * 4 - TempLen$$

where

$$TempLen = 32 + L\_N + L\_D$$

2.6.13 Track Indexes

This field shall contain an array of track indexes that form the playlist.

2.6.14 List Padding

This field shall contain padding bytes to ensure that the *Extra Data* field starts at 4–byte alignment boundary. The length of that field is

$$int((TempLen + 3)/4) * 4 - TempLen$$

where

$$TempLen = 32 + L\_N + L\_D + L\_Pp + 2 * N\_T$$

2.6.15 Extra Data

A *Chunk* containing an array of *Chunk* based structures (Section 2.9).

2.7 Tracklist format

Tracklist structure is an equivalent of a *Playlist* structure but does not contain references (indexes, etc.) to any `TOC.MAU` structure. Instead of indexes it contains full *TrackEntry* structures. It is placed in a separate file. The purpose of this this is to allow low memory devices to discard the "table of contents" information read previously.

Note: *Each Tracklist shall have an equivalent Playlist and vice versa.*

#	RBP	Length	Name	Contents
1	0	12	Tag	`tag`
2	12	2	Number of Tracks (=N_T)	`Uint16`
3	14	2	Text Format	`Uint16`
4	16	2	Offset to Name	`Uint16`
5	18	2	Offset to Description	`Uint16`
6	20	2	Offset to Padding	`Uint16`
7	22	2	Offset to Track Indexes	`Uint16`
8	24	4	Offset to List Padding	`Uint32`
9	28	4	Offset to Extra Data	`Uint32`
10	32	L_N	Playlist Name	`byte`
11	$32 + L\_N$	L_D	Playlist Description	`byte`
12	$32 + L\_N + L\_D$	L_Pp	Padding	`byte`
13	$32 + L\_N + L\_D + L\_Pp$	L_Tr	Track Entries	`TrackEntry`
14	$32 + L\_N + L\_D + L\_Pp + L\_Tr$	L_Pp	List Padding	`byte`
15	$32 + L\_N + L\_D + L\_Pp + L\_Tr + L\_Pp$	L_Ex	Extra Data	`ExtraData`

Note: *Providing information in all the fields is mandatory, unless stated otherwise in a field definition.*

2.7.1 Tag

`Tag.Identifier` subfield shall be set to 54524C53h.

`Tag.OrdinalNumber` subfield shall be set to the 0–based number of the playlist in the list of playlists.

`Tag.Length` subfield shall be set to the length of this *Playlist* structure.

2.7.2 Number of Tracks

This field shall be set to the number of tracks in the play list.

2.7.3 Text Format

If this field is set to 0 then *Play List Name* and *Play List Description* shall be interpreted as *ASCII* strings.

If this field is set to 1 then all *Play List Name* and *Play List Description* shall be interpreted as *UNICODE* strings.

2.7.4 Offset to Name

This field shall be set to the offset (in bytes, from the beginning of *Playlist*) to the *Playlist Name* string.

2.7.5 Offset to Description

This field shall be set to the offset (in bytes, from the beginning of *Playlist*) to the *Play List Description* string.

2.7.6 Offset to Padding

This field shall be set to the offset (in bytes, from the beginning of *Playlist*) to the padding before the array of track indexes. If the number of padding bytes is zero (padding not present) then the offset to padding shall be set to zero.

2.7.7 Offset to Track Indexes

This field shall be set to the offset (in bytes, from the beginning of *Playlist*) to track entry index array.

2.7.8 Offset to List Padding

This field shall be set to the offset (in bytes, from the beginning of *Playlist*) to the padding before *Extra Data* field. If the number of padding bytes is zero (padding not present) then the offset to padding shall be set to zero.

2.7.9 Offset to Extra Data

This field shall be set to the offset (in bytes, from the beginning of *Playlist*) to the *Extra Data* field. If the *Extra Data* field is not provided then this field shall be set to zero.

2.7.10 Playlist Name

This field shall contain the playlist name according to the character set defined by *Text Format* field.

2.7.11 Playlist Description

This field shall contain the play list description according to the character set defined by *Text Format* field.

2.7.12 Padding

This field shall contain padding bytes to ensure that the array of track indexes starts at 4–byte alignment boundary. The length of that field is

$$int((TempLen + 3)/4) * 4 - TempLen$$

where

$$TempLen = 32 + L\_N + L\_D$$

2.7.13 Track Entries

This field shall contain an array of *TrackEntry* (2.5) that form the playlist.

2.7.14 List Padding

This field shall contain padding bytes to ensure that the *Extra Data* field starts at 4–byte alignment boundary. The length of that field is

$$int((TempLen + 3)/4) * 4 - TempLen$$

where

$$TempLen = 32 + L\_N + L\_D + L\_Pp + L\_Tr$$

Note: *Since TrackEntry structures are guaranteed to be evenly divisible by 4, the number of padding bytes will be always zero. This field and the offset to it is present to maitain the similarity to the Playlist structure.*

2.7.15 Extra Data

A *Chunk* containing an array of *Chunk* based structures (Section 2.9).

2.8 PlaylistDirectory

The *PlaylistDirectory* structure allows for grouping playlists.

#	RBP	Length	Name	Contents
1	0	12	Tag	`tag`
2	12	2	Number of Playlists ($=N\_P$)	`Uint16`
3	14	2	Text Format	`Uint16`
4	16	2	Offset to Name	`Uint16`
5	18	2	Offset to Description	`Uint16`
6	20	2	Offset to Pathname CSD	`Uint16`
7	22	N_P*2	Playlist Indexes	`Uint16`
8	$22+N\_P*2$	N_P*2	Offsets to Tracklist Pathnames	`Uint16`
9	$22+N\_P*2+N\_P*2$	2	Offset to Padding	`Uint16`
10	$24+N\_P*2+N\_P*2$	2	Offset to Extra Data	`Uint16`
11	$26+N\_P*2+N\_P*2$	L_N	Name	`byte`
12	$26+N\_P*2+N\_P*2+L\_N$	L_D	Description	`byte`
13	$26+N\_P*2+N\_P*2+L\_N+L\_D$	L_LS	Tracklist Pathnames	`byte`
14	$26+N\_P*2+N\_P*2+L\_N+L\_D+L\_LS$	L_P	Padding	`byte`
15	$26+N\_P*2+N\_P*2+L\_N+L\_D+L\_LS+L\_P$	L_Cs	Pathname CSD	`byte`
16	$26+N\_P*2+N\_P*2+L\_N+L\_D+L\_LS+L\_P+L\_Cs$	L_Ex	Extra Data	`ExtraData`

Note: *Providing information in all the fields is mandatory, unless stated otherwise in a field definition.*

2.8.1 Tag

`Tag.Identifier` subfield shall be set to 4.

`Tag.OrdinalNumber` subfield shall be set to the 0–based number of the playlist directory in the list of playlist directories.

`Tag.Length` subfield shall be set to the length of this *PlaylistDirectory* structure.

2.8.2 Number of Playlists

This field shall be set to the number of playlists in the directory.

2.8.3 Text Format

If this field is set to 0 then text fields except for pathnames shall be interpreted as *ASCII* strings.

If this field is set to 1 then all text fields except for pathnames shall be interpreted as *UNICODE* strings.

2.8.4 Offset to Name

This field shall be set to the offset to description (in bytes, from the beginning of *Playlist-Directory*) of the directory.

2.8.5 Offset to Description

This field shall be set to the offset to description (in bytes, from the beginning of *Playlist-Directory*) of the directory.

2.8.6 Offset to Pathname CSD

This field is provided for use by filesystems that might use more than one character set for describing pathnames. CSD stands for Character Set Descriptor. Starting from the version 1.10 this is a required field. See Section 4.1 for more information.

2.8.7 Playlist Indexes

This field shall be set to indexes (i.e. 0–based consecutive playlist numbers) representing playlists within *TOC*. Each index shall correspond to a tracklist name.

2.8.8 Offsets to Tracklist Pathnames

This field shall contain a list of offsets (in bytes, from the beginning of *PlaylistDirectory*) to tracklist pathnames.

2.8.9 Offset to Padding

This field shall be set to the offset (in bytes, from the beginning of *PlaylistDirectory*) to the padding before *Extra Data* field. If the number of padding bytes is zero (padding not present) then the offset to padding shall be set to zero.

2.8.10 Offset to Extra Data

This field shall be set to the offset (in bytes, from the beginning of *PlaylistDirectory*) to the *Extra Data* field. If the *Extra Data* field is not provided then this field shall be set to zero.

2.8.11 Name

This field shall contain the name of the directory, for example "`Jazz songs`". The field shall be encoded according to the value of *Text Format* field.

2.8.12 Description

This field shall contain the (more verbose than the name) description of the directory, for example "`Jazz songs, good old standards.`". The field shall be encoded according to the value of *Text Format* field.

2.8.13 Tracklist Pathnames

The pathnames are relative to the *MA* `TOC.MAU` file. Depending on the value of *Pathname CSD*, the format of a pathname field is defined by the *Text Format* field (Section 2.8.3), **or** it is stored in the "native" format. Detailed explanation of how to inpterpret *Pathname CSD* can be found in Section 4.1.

2.8.14 Padding

Padding to ensure that *Extra Data* starts at 4–byte aligned boundary. The length of that field is

$$int((TempLen + 3)/4) * 4 - TempLen$$

where

$$TempLen = 26 + N\_P * 2 + N\_P * 2 + L\_N + L\_D + L\_LS$$

2.8.15 Pathname CSD

The Pathname Character Set Descriptor (CSD) is a required field that is compatible with the Chunk structure (Section 2.3). The CSD is used to interpret the format of the Pathname field (Section 2.8.13. The definition of the contents of this structure can be found in Section 4.1.

2.8.16 Extra Data

A *Chunk* containing an array of *Chunk* based structures (Section 2.9).

2.9 ExtraData format

#	RBP	Length	Name	Contents
1	0	12	Tag	`tag`
2	12	*L_C*	Array of Chunks	`Chunk`

2.9.1 Tag

`Tag.Identifier` subfield shall be set to 5.

`Tag.OrdinalNumber` subfield shall be set to 0.

`Tag.Length` subfield shall be set to the length of *ExtraData* structure, which is equal to the length of the *Tag* field plus lengths of *Chunks* contained in the data area.

2.9.2 Array of Chunks

Extra Data contents is an array of *Chunks* (Section 2.3), which shall be ordered by the value

of the `Tag.Identifier` field.

Note: *This will place MA structures first, other "well-known" data types next, and user/application private extensions last, in the Extra Data area.*

The `Tag.OrdinalNumber` field of each *Chunk* has to be set to a number representing occurrence of a particular data type in a particular *Extra Data* field. For example, if an *Extra Data* field contains m *Chunks* of type X and n *Chunks* of type Y, then `Tag.OrdinalNumber` fields of X records are set to values from 0 to $m-1$ and the `Tag.OrdinalNumber` fields of Y records are set to values from 0 to $n-1$.

3 Information Layout on a MultiAudio media

The overall *MA* layout is shown in Figure 1.

3.1 MultiAudio Files

MultiAudio files are:

- `TOC.MAU` - this file contains the MultiAudio table of contents (Section 3.2). `TOC.MAU` shall be placed in the root directory of a *MA* media.

- Tracklists - these files contain track entries corresponding to user defined playlists (Section 2.7). *Each* user defined playlist shall have a corresponding tracklist. The standard does not place any restrictions on the location of tracklist files. As shown in Figure 1 playlist directories contain links (i.e. pathnames) to tracklist files.

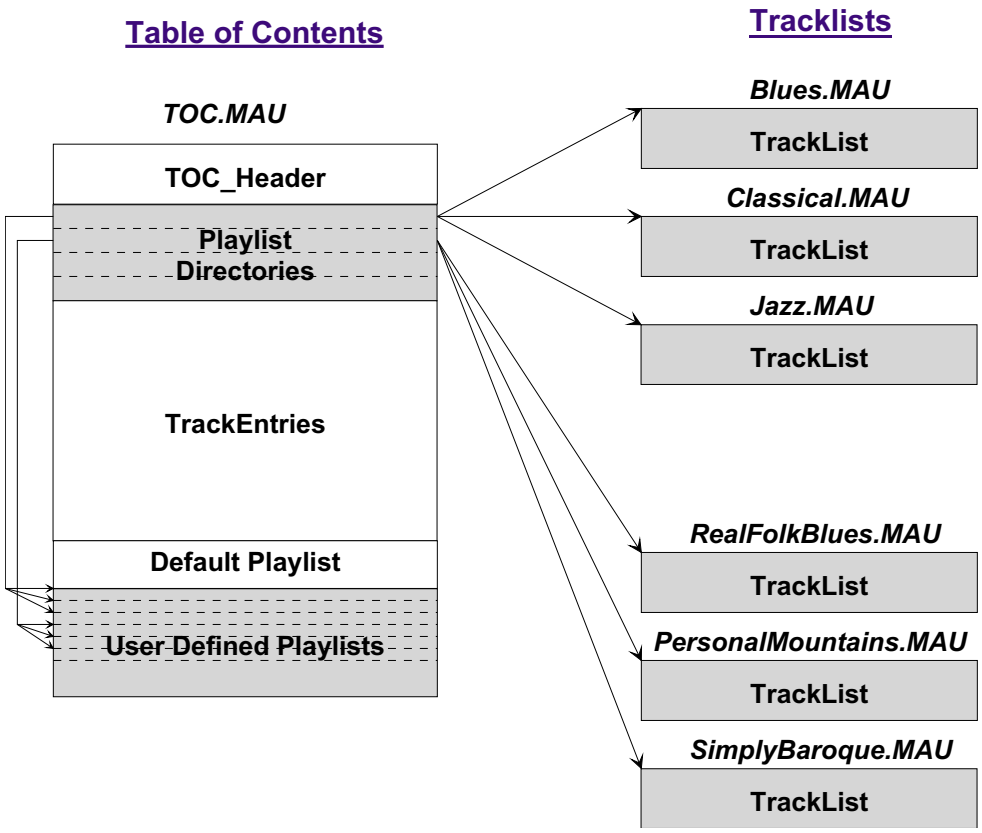

Figure 1: *MultiAudio* data layout.

3.2 TOC

The *TOC* layout is as follows

- it shall contain one header,
- it may contain a list of playlist directories,
- it shall contain a list of tracks,
- it shall contain a default playlist,
- it may contain more user defined playlists.

A diagram shown in Figure 1 illustrates the layout. Grayed items are optional.

The default playlist shall contain all the tracks in the order they were recorded (the order of track entries in *TOC*).

3.3 Playlist Directories

Playlist directories serve two important purposes:

1. A *PlaylistDirectory* provides a link between a playlist and an equivalent tracklist.
2. They allow for organizing playlists according to genres, themes, etc.

Note: *There has to be at least one PlaylistDirectory if there are user defined playlists. This is because of the consistency requirement that playlists have equvalent tracklists and the fact that only a PlaylistDirectory provides links the playlist to the tracklist.*

3.4 Tracklists

Tracklists are an optional form of representing playlist for low memory devices. The idea is that a device would load only the *TOC_Header* and playlist directories. These directories contain links to files that consist of *TrackEntries* representing specific playlists.

Note: *The easy access to tracklists is the main reason for the tracklist pathname to be present in a PlaylistDirectory instead of a Playlist – which might seem more natural. But if the tracklist pathname was located in a Playlist structure then a device would have to access the Playlist in addition to TOC_Header and playlist directories.*

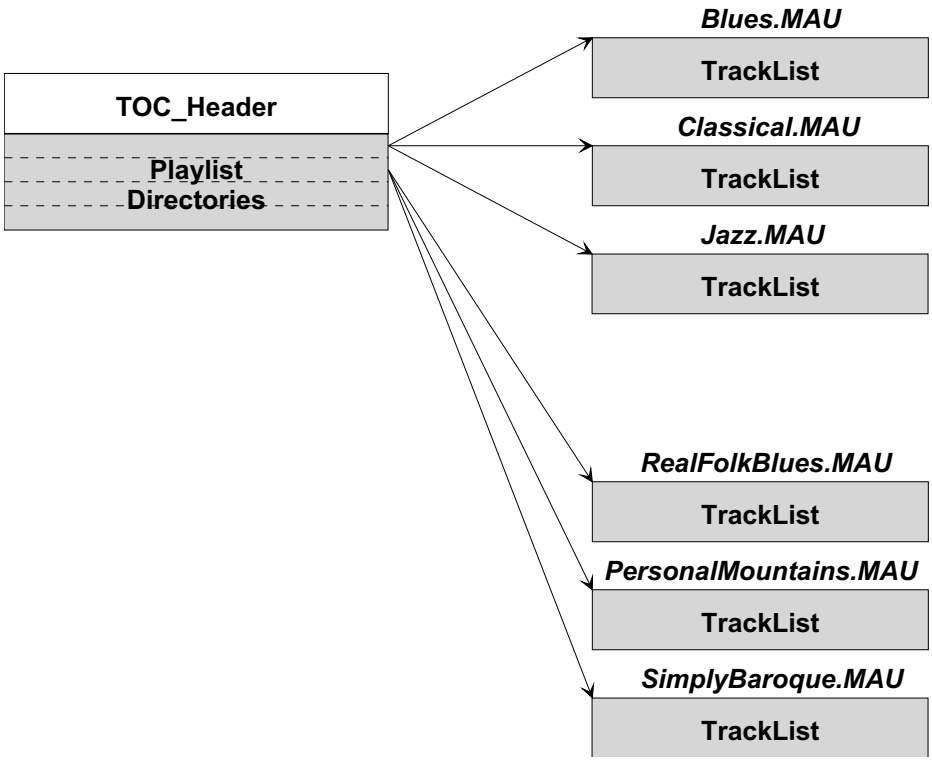

Figure 2: Concept of tracklists.

After selecting the tracklist file, the device may discard *TOC* information and load just the tracklist.

Note: *The same track entry may appear in many tracklists.*

4 Notes

4.1 Pathname Character Set Descriptor

The Pathname Character Set Descriptor (CSD) is defined as a *Chunk* structure (see Section 2.3). It is a required field that is used to interpret the format of the pathname field defined in the different MultiAudio structures. The chunk structure defined in this field can have 2 possible values for the Identifier entry in its Tag field. They are the following:

Tag.Identifier	Pathname Encoding	Chunk Length	Chunk Data
00010000h	File System Specific ("native") Encoding	Variable	Filesystem Specific
00010001h	Defined by *Text Format* field (ASCII Encoding or Unicode Encoding)	12	–

The native representation (identifier equal to 00010000h) is described by the Section 4.2. For this representation, the Pathname CSD will contain the filesystem specific information in the data area.

Encoding dependent on the value of *Text Format* field means that the pathname must be encoded just like all the other strings in a particular structure. Note that each Track Entry structure and Playlist Directory structure has *Text Format* field. For this represenation, the chunk will contain no data and the tag length value will contain 12 which is the length of the chunk header only. The non–native pathname representation is described in Section 4.3.

4.2 Native Pathname Representation

The File System Specific (native) encoding is the most robust form of pathname encoding designed to work with all possible file systems that the `TOC.MAU` file might be present on. Since every file system has its own unique format and limitations, to ensure that the file can be accurately located, it might be necessary to encode the pathname in the format recognized by file system in which the TOC file is located. For example, if the TOC file is on an ISO9660 file system, the pathname must be in ASCII format with an 8 characters for the filename and three characters for the file extension. Alternatively, if the TOC file is on a Joliet file system, the pathname must encoded in MSB UNICODE format, and can only be a max of 64 characters long. In the case of a file system that allows many character sets to store a pathname, such as UDF, the chunk data would contain a character set descriptor that defines the character set used. An example is shown below:

OSTA CS0 Charspec

```
struct Charspec
{
  Uint8 CharacterSetType;
  byte CharacterSetInfo[63];
}
```

The `CharacterSetType` field shall have the value of 0 to indicate the `CS0` coded character set.
The `CharacterSetInfo` field shall contain the following byte values with the remainder of the field set to a value of 0: 4Fh, 53h, 54h, 41h, 20h, 43h, 6Fh, 6Dh, 70h, 72h, 65h, 73h, 73h, 65h, 64h, 20h, 55h, 6Eh, 69h, 63h, 6Fh, 64h, 65h
The above byte values represent the following ASCII string: "OSTA Compressed Unicode"

Directory separator shall be either backward slash (\) or forward slash (/) for **all** *MA* pathnames within a particular filesystem namespace. This means that interchanging forward and backward slash accross *MA* files and structures is not allowed.

Pathname shall **not** be terminated by any null characters. The length of a pathname string should be derived from the length of appropriate structure and offset values.

4.3 Non–Native Pathname Representation

Storing pathnames using ASCII and UNICODE encoding will require using software translating these strings to the file system native representation. For this reason a pathname shall not use characters that cannot be translated by "fitting into the wider character" conversion or lower–case/upper–case conversion. Using characters that have to be simply expanded into wider characters (by filling with zeros for example) is permitted. Using illegal characters for a given file system that have to be translated to other characters is not permitted. Using characters that would have to be truncated and thus misrepresented is not permitted (in other words we can only truncate a filling). If the pathname contains illegal characters, or cannot be translated to a file system native representation, it is recommended that the File System Specific format be used.

As in case of native representation, directory separator shall be either backward slash (\) or forward slash (/) for **all** *MA* pathnames within a particular filesystem namespace. This means that interchanging forward and backward slash accross *MA* files and structures is not allowed.

For example, let's consider ISO9660 file format. An ISO9660 filename must be in the form 8+3, which is 8 characters for the filename, and 3 characters for the file extension. Allowed character set is A–Z plus underscore. Examples of legal pathnames are: *`MEDIA\SONG.MP3`*, *`Media\Song2.MP3`*. Examples of illegal pathnames:

MEDIA\SONG-1.MP3 – Illegal character '-' is used.

Media\MyFavoriteSong.mp3 – Too long filename, there are many ways to convert it to an ISO9660 compatible filename.

4.4 Filesystem Namespaces

It is possible that the media contains more than one filesystem (it is a so called "bridge disc"). Most likely different filesystems will have different restrictions on pathname representation. For example the ISO9660 filesystem allows for a very limited character set and short filenames. Various extensions to ISO9660 and UDF include support for wide characters and longer names.

The specification requires, that each filesystem stores a "its own" set of *MA* files (*TOC.MAU* and user defined tracklists) containing pathnames in its "native" format.

For example an ISO9660/Joliet disc would contain two *TOC.MAU* files. One would be visible by ISO9660 only, the other would be visible ty Joliet only.

The same principle would apply for other bridge discs (ISO9660/UDF, etc.).

5 Revision History

5.1 1.00 → 1.10 (September 16, 2002)

- The specification version changed to 1.1.0.
- The *Pathname CSD* field is now mandatory. This field specifies whether the pathname is represented in the "native" format (the only allowed format so far) or as other strings in a given structure. This change is reflected in Sections: 2.5.9, 2.5.33, 2.5.35, 2.8.6, 2.8.13, 2.8.15.
- The "Notes" section was added. This section clarifies confusing issues like representation of pathnames.

5.1.1 TOC_Header Clarifications

- Section 2.4.2: the "Version Number" field was clarified. Previous contents:

  This field shall contain the MA standard version number. The version defined by this document is 1.00.
- Section 2.4.4: description of the "Length of TOC" field was clarified. Previous contents:

  Length of the entire CA Table of Contents structure in bytes.
- Section 2.4.6: description of the "Volume Name" field was clarified. Previous contents:

  This field may contain a user specified disc/volume name. The string shall be encoded according to Text Format (2.4.5).

5.1.2 TrackEntry Clarifications

- Section 2.5.25: description of the "Encoding TID Padding" field was clarified. Previous contents:

  This field provides a 2–byte alignment of the Track Name. Since the encoding type identifier is always a d–string *it could cause a misalignment of a track name stored in UNICODE (Text Format = 1). For ASCII strings the length of this field is always zero. Computing the length of the padding field is described by the formula.*

  $$TextFormat * rem(L\_Et, 2)$$

  In other words, if Text Format is 0 then the length of the padding is zero. If Text Format is 1 (UNICODE strings) then the padding field is 1 if the L_Et is odd and 0 if L_Et is already even.

- Section 2.5.33: description of the "Pathname" field was clarified. Previous contents:

  This field shall contain the pathname encoded in a format supported by the filesystem that contains these MA structures.

- Section 2.5.11: description of the "Offset to Encoding TID Padding" field was clarified. Previous contents:

  This field shall be set to the offset (in bytes, from the beginning of TrackEntry) to the encoding type identifier padding. TID stands for Type Identifier.

- Section 2.5.20: description of the "Offset to Pathname Padding" field was clarified. Previous contents:

- *Section 2.5.11: description of the "Offset to Encoding TID Padding" field was clarified. Previous contents:*

5.1.3 Playlist Directory Clarifications

- Section 2.8.13: description of the "Tracklist Pathnames" field was clarified. Previous contents:

  This field shall contain pathnames.The pathnames shall be encoded according to the default character set for a given filesystem or according to the specified character set.

- Section 2.8.9: description of the field "Offset to Padding" was clarified: Previous contents:

  This field shall be set to the offset (in bytes, from the beginning of PlaylistDirectory) to the padding before Extra Data field.

5.2 0.73 → 1.00 (July 25, 2001)

- The version number was modified from 0.73 to 1.00.
- Figures used in the document were modified to reflect changes in filenames (`CATOC.DAT` to `TOC.MAU`).
- Remaining occurrences of `CATOC.DAT` in the document were changed to `TOC.MAU`.
- *TOC_Header* UUID field description was clarified.
- The *Tag* structure was modified to expand the length field to 32 bits (this allows for large *Chunk* structures, important for *Tracklist* files.
- The *Playlist* and *Tracklist* structures were modified:

1. *Name* and *Description* fields were moved in front of track entry index array for *Playlist* and in front of track entries for *Tracklist* file. This way offsets to these fields can stay 16–bit long for large tracklist files.
2. Offset to *Extra Data* field was expanded to 32 bits to allow for large tracklist files.
3. Offset to track indexes/track entries was added as a consequence of the first change.
4. *Padding2* field was added in front of the *Extra Data* field (consequence of the first change too).
5. Offset to the *Padding2* was added.

- PDF bookmarks to appendices and index were added.

5.3 0.72 → 0.73 (July 11, 2001)

- Appendix B containing the definition of UUID string was added.

5.4 0.71 → 0.72 (June 20, 2001)

- The name of the specification changed from *Compressed Audio* to *MultiAudio*.
- the name of table of contents file changed from *CATOC.DAT* to *TOC.MAU*.
- Definition of *DateAndTime* structure was added.
- UUID string replaced *Signature* in the *TOC_Header* structure.
- Timestamp fields (*Creation Date And Time*, *Modification Date And Time*, *Effective Date And Time*, *Expiration Date And Time*) and identifier fields (*Data Preparer Identifier*, *Publisher Identifier*, *Copyright*) were added to *TOC_Header*.
- More exposed statement in the Section 2.3
- *ExtraData* type was added and defined as a *Chunk* encapsulating an array of *Chunks*. that a *Chunk* has to be evenly divisible by four was added.
- More verbose explanation that offsets to optional fields shall be zero if these fields are not provided was added.
- Space for *MA* files in the *Tag* Identifier namespace was reserved.
- WAV to "well known" encoding list was added.
- The field *Name* to *PlaylistDirectory* was added.

- The field *Playlist Descriptions* was removed from *PlaylistDirectory*. This field is not necessary, as *Traclists* will now contain the same fields as *Playlists*.

5.5 0.70 → 0.71 (April 25, 2001)

- Some entries in the *TrackEntry* and the *PlaylistDirectory* were shortened. "Encoding Type Identifier" became "Encoding TID" and "Pathname Character Set Descriptor" became "Pathname CSD". Abbreviations were explained in the relevant field description sections.

5.6 0.69 → 0.70 (April 13, 2001)

- The 0.69 version of *CA* specification had a wrong formula for computing of a padding size.

5.7 0.68 → 0.69 (April 12, 2001)

- Specification name: "Compressed Digital Audio" changed to "Compressed Audio". Consequently, *CDA* prefixes changed to *CA*.

- ISO9660 pathnames (as well as all the references to a specific filesystem) were removed from *CA* structures. This includes ISO9660 and UDF support flags in the `Flags` field in *TOC_Header* structure (Section 2.4.19). Right now the specification requires that any filesystem that supports *CA* must duplicate *CA* files containing proper pathnmes in its namespace.

- *TracklistDirectory* changed name to *PlaylistDirectory* (Section 2.8). We require now that there is playlist representation consistency, i.e. each playlist within a *CATOC* has to have equivalent tracklist file on a *CA* disc.

5.8 0.67 → 0.68 (March 29, 2001)

- A UDF support flag was defined for the `Flags` field in *TOC_Header* structure (Section 2.4.19).

- Integer fields were defined as recorded in Little Endian format (Section 1.4.3).

- Definitions of basic data types were added (Section 1.4.3).

- A *CDATOC* prefix was removed from names of structures (better readablility). Thus, *CDATOC_TrackEntry* becomes *TrackEntry*, *CDATOC_Playlist* becomes *Playlist* and *CDATOC_TracklistDirectory* becomes *TracklistDirectory*. The *CDATOC_Header* was changed to *TOC_Header*.
- Field names like *Track Count* were changed to *Number of Tracks*.
- The *TOC_Header* structure now contains an offset to the *Extra Data* field instead of it's length. That was done for the consistency with other structures.

5.9 0.66 → 0.67 (March 20, 2001)

- The string definition was changed to distinguish between a character set used for ISO9660 pathnames and character set used for other string fields.
- *CDATOC* structures were modified to be more convenient for programmers. Specifically, offsets to string fields were added which doesn't require tedious computing of a string location.
- The layout of CDA information was clarified, the concept of tracklists was described in more detail.

5.10 0.65 → 0.66 (March 13, 2001)

- More tags to *CDATOC_TrackEntry* were added.
- The *Encoding Type* field of *CDATOC_TrackEntry* was modified from an integer into variable length literal representation.
- A signature field was added to the *CDATOC_Header*.

5.11 0.64 → 0.65 (March 12, 2001)

- No more support for physical addressing.
- No more signature in sector 16.

5.12 0.63 → 0.64

- The document was completely reformatted.

5.13 0.62 → 0.63

- Added the Volume field in *CDATOC_Header*.

5.14 0.60 → 0.61

- Added ISO9660 support flag to *CDATOC_Header*.
- Added long filename fields to *CDATOC_TrackEntry*.
- Added Song year and Song order to *CDATOC_TrackEntry*.
- Now require a minumum eof three playlists.
- Genre playlist extra data defined.

5.15 0.55 → 0.60 (January 31, 2001)

- Added structure length field to *CDATOC_Header*.
- Changed allocation for track and playlist indexes from fixed to variable in *CDATOC_Header*.
- Added extra data count and field in *CDATO_Header* for future additions to the specification.
- Removed track description field from *CDATOC_TrackEntry*.
- Added extra data field to *CDATOC_TrackEntry*.
- Defined file name field in *CDATOC_TrackEntry* to always be ASCII 8.3 ISO9660 compatible.
- Added additional compressed audio types to *CDATOC_TrackEntry*.
- Added extra data field to *CDATOC_Playlist*
- Changed PVD signature to eliminate version number, use version in the *CDATOC_Header*.
- Changed version in *CDATOC_Header* to 60

Appendix A – Encoding Types

The "well known" *Encoding Type Identifiers* are:

String	Hex representation	Interpretation
UNKNOWN	55 4e 4b 4e 4f 57 4e 00	Unknown encoding
MP3	4d 50 33 00	Digital Audio file compressed from PCM (RAW) form using MPEG1, Layer 3 method.
WMA	57 4d 41 00	Microsoft Windows Media Audio file – Digital Audio file compressed from PCM (RAW) form.
WAV	57 41 56 00	Microsoft uncompressed audio file
ATRAC3	41 54 52 41 43 4b 33 00	Adaptive Transform Acoustic Coding 3
MPEG2_AAC	4d 50 45 47 32 5F 41 41 43 00	MPEG–2 Advanced Audio Coding - Successor to MPEG-1 Layer-3 compression.
MPEG4_AAC	4d 50 45 47 34 5F 41 41 43 00	MPEG–4 Advanced Audio Coding - Has all the features of MPEG–2 ACC plus Perceptual Noise Substitution (PNS), a Long Term Predictor (LTP) and extensions to support scalability.
TWINVQ	54 57 49 4e 56 51 00	Transform-Domain Weighted Interleave Vector Quantization
OGG_VORBIS	4F 47 47 5F 56 4F 52 42 49 53 00	Open source, patent-free, royalty-free audio codec.

Identifiers, just like all the strings shall be null-terminated.

UNKNOWN encoding type means that it is up to the device to determine the type.

"Private" identifiers are allowed but must be prefixed with `X-`.

Note: *This appendix should eventually be maintained as a separate document. Changes to the "well known" encoding type identifier list are more related to development of new encoding types in the industry and do not affect the structure of the specification.*

Appendix B – UUID String Format Specification

A universally unique identifier (UUID) format was defined in the Open Software Foundation's Distributed Computing Environment RPC standard also available as ISO-11578, which defines UUIDs in an appendix.

A internet draft was proposed that specifically defines UUIDs. This expired in 1998 and was removed from the standard location at http://search.ietf.org/internet-drafts/draft-leach-uuids-guids-01.txt. Various copies still exist on the internet and are useful defacto standards. http://www.ics.uci.edu/pub/ietf/webdav/uuid-guid/draft-leach-uuids-guids-01.txt This draft standard also includes source code for UUID generation both with and without the use of ethernet MAC addresses.

The relevant excerpt for UUID string representation, UUID comparison and copyright cited after Network Working Group Internet Draft is as follows:

```
3.5 String Representation of UUIDs

  For use in human readable text, a UUID string representation is
  specified as a sequence of fields, some of which are separated by
  single dashes.

  Each field is treated as an integer and has its value printed as a
  zero-filled hexadecimal digit string with the most significant digit
  first. The hexadecimal values a to f inclusive are output as lower
  case characters, and are case insensitive on input. The sequence is
  the same as the UUID constructed type.

  The formal definition of the UUID string representation is provided
  by the following extended BNF:

  UUID                   = <time_low> "-" <time_mid> "-"
                           <time_high_and_version> "-"
                           <clock_seq_and_reserved>
                           <clock_seq_low> "-" <node>
  time_low               = 4*<hexOctet>
  time_mid               = 2*<hexOctet>
  time_high_and_version  = 2*<hexOctet>
  clock_seq_and_reserved = <hexOctet>
  clock_seq_low          = <hexOctet>
  node                   = 6*<hexOctet
  hexOctet               = <hexDigit> <hexDigit>
  hexDigit =
        "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
        | "a" | "b" | "c" | "d" | "e" | "f"
        | "A" | "B" | "C" | "D" | "E" | "F"
```

The following is an example of the string representation of a UUID:

f81d4fae-7dec-11d0-a765-00a0c91e6bf6

3.6 Comparing UUIDs for equality

Consider each field of the UUID to be an unsigned integer as shown in the table in section 3.1. Then, to compare a pair of UUIDs, arithmetically compare the corresponding fields from each UUID in order of significance and according to their data type. Two UUIDs are equal if and only if all the corresponding fields are equal.

Note: as a practical matter, on many systems comparison of two UUIDs for equality can be performed simply by comparing the 128 bits of their in-memory representation considered as a 128 bit unsigned integer. Here, it is presumed that by the time the in-memory representation is obtained the appropriate byte-order canonicalizations have been carried out.

3.7 Comparing UUIDs for relative order

Two UUIDs allocated according to the same variant can also be ordered lexicographically. For the UUID variant herein defined, the first of two UUIDs follows the second if the most significant field in which the UUIDs differ is greater for the first UUID. The first of a pair of UUIDs precedes the second if the most significant field in which the UUIDs differ is greater for the second UUID.

10. Notice

11. Full Copyright Statement

Index