# INTERNATIONAL STANDARD

# ISO/IEC 14496-12

Fourth edition

2012-07-15

Corrected version

2012-09-15



**Information technology — Coding of**

**audio-visual objects —**

Part 12:

**ISO base media file format**

*Technologies de l'information — Codage des objets audiovisuels —*

*Partie 12: Format ISO de base pour les fichiers médias*

目录

[INTERNATIONAL STANDARD 1](#_Toc24945)

[ISO/IEC 14496-12 1](#_Toc1262)

[Foreword 5](#_Toc11192)

[Introduction 6](#_Toc4295)

[1 Scope 7](#_Toc28103)

[2 Normative references 7](#_Toc10478)

[3 Terms, definitions, and abbreviated terms 8](#_Toc23796)

[3.1 Terms and definitions 8](#_Toc15515)

[3.2 Abbreviated terms 10](#_Toc26096)

[4 Object-structured File Organization 10](#_Toc7882)

[4.1 File Structure 10](#_Toc27809)

[4.2 Object Structure 11](#_Toc5404)

[4.3 File Type Box 12](#_Toc13337)

[5 Design Considerations (注意事项) 13](#_Toc25223)

[5.1 Usage 13](#_Toc30454)

[6 ISO Base Media File organization 15](#_Toc21680)

[6.1 Presentation structure 15](#_Toc15328)

[6.2 Metadata Structure (Objects) 16](#_Toc20706)

[6.3 Brand Identification 20](#_Toc3861)

[7 Streaming Support 21](#_Toc1540)

[7.1 Handling of Streaming Protocols 21](#_Toc20757)

[7.2 Protocol ‘hint’ tracks 21](#_Toc21243)

[7.3 Hint Track Format 22](#_Toc3053)

[8 Box Structures 23](#_Toc11608)

[8.1 File Structure and general boxes 23](#_Toc3005)

[8.2 Movie Structure 25](#_Toc29259)

[8.3 Track Structure 26](#_Toc7263)

[8.4 Track Media Structure 31](#_Toc7198)

[8.5 Sample Tables 35](#_Toc7517)

[8.6 Track Time Structures 43](#_Toc26843)

[8.7 Track Data Layout Structures 53](#_Toc26271)

[8.8 Movie Fragments 62](#_Toc8900)

[8.9 Sample Group Structures 73](#_Toc2933)

[8.10 User Data 76](#_Toc440)

[8.11 Metadata Support 79](#_Toc14046)

[8.12 Support for Protected Streams 90](#_Toc5326)

[8.13 File Delivery Format Support 93](#_Toc29514)

[8.14 Sub tracks 99](#_Toc23970)

[8.15 Post-decoder requirements on media 102](#_Toc294)

[8.16 Segments 105](#_Toc21287)

[9 Hint Track Formats 111](#_Toc15080)

[9.1 RTP and SRTP Hint Track Format 111](#_Toc28676)

[9.2 ALC/LCT and FLUTE Hint Track Format 118](#_Toc14981)

[9.3 MPEG-2 Transport Hint Track Format 123](#_Toc20665)

[9.4 RTP, RTCP, SRTP and SRTCP Reception Hint Tracks 130](#_Toc21352)

[10 Sample Groups 138](#_Toc18473)

[10.1 Random Access Recovery Points 138](#_Toc19821)

[10.2 Rate Share Groups 139](#_Toc17723)

[10.3 Alternative Startup Sequences 142](#_Toc28783)

[10.4 Random Access Point (RAP) Sample Grouping 145](#_Toc31404)

[10.5 Temporal level sample grouping 146](#_Toc11611)

[11 Extensibility 146](#_Toc586)

[11.1 Objects 146](#_Toc22775)

[11.2 Storage formats 147](#_Toc16176)

[11.3 Derived File formats 147](#_Toc19895)

[Annex A 148](#_Toc7868)

[A.1 Section Overview 148](#_Toc13650)

[A.2 Core Concepts 148](#_Toc5714)

[A.3 Physical structure of the media 149](#_Toc32763)

[A.4 Temporal structure of the media 149](#_Toc27697)

[A.5 Interleave 149](#_Toc32363)

[A.6 Composition 150](#_Toc3036)

[A.7 Random access 150](#_Toc32024)

[A.8 Fragmented movie files 150](#_Toc27370)

[Annex B 151](#_Toc29264)

[Annex C 152](#_Toc26939)

[C.1 Introduction 152](#_Toc7702)

[C.2 General Principles 152](#_Toc12255)

[C.3 Boxes 153](#_Toc12781)

[C.4 Brand Identifiers 154](#_Toc27700)

[C.5 Storage of new media types 155](#_Toc2042)

[C.6 Use of Template fields 156](#_Toc28939)

[C.7 Tracks 156](#_Toc27674)

[C.8 Construction of fragmented movies 158](#_Toc21970)

[C.9 Meta-data 159](#_Toc20204)

[C.10 Registration 159](#_Toc11275)

[C.11 Guidelines on the use of sample groups, timed metadata tracks,and sample auxiliary information 159](#_Toc14809)

[Annex D 160](#_Toc26019)

[D.1 Code points to be registered 160](#_Toc21657)

[D.2 Procedure for the request of an MPEG-4 registered identifier value 161](#_Toc6887)

[D.3 Responsibilities of the Registration Authority 161](#_Toc4056)

[D.4 Contact information for the Registration Authority 161](#_Toc5412)

[One Infinite Loop, M/S 301-4B 161](#_Toc26071)

[D.5 Responsibilities of Parties Requesting a RID 162](#_Toc27440)

[D.6 Appeal Procedure for Denied Applications 162](#_Toc26942)

[D.7 Registration Application Form 162](#_Toc10105)

[Annex E 163](#_Toc23431)

[E.1 Introduction 163](#_Toc27467)

[E.2 The ‘isom’ brand 164](#_Toc4004)

[E.3 The ‘avc1’ brand 165](#_Toc24929)

[E.4 The ‘iso2’ brand 165](#_Toc17088)

[E.5 The ‘mp71’ brand 166](#_Toc8716)

[E.6 The ‘iso3’ brand 166](#_Toc9312)

[E.7 The ‘iso4’ brand 166](#_Toc32106)

[E.8 The ‘iso5’ brand 167](#_Toc20522)

[E.9 The ‘iso6’ brand 167](#_Toc1159)

[Annex F 167](#_Toc20234)

[Annex G 169](#_Toc12385)

[G.1 UUID-labelled metadata 169](#_Toc8871)

[G.2 ISO OID-labelled metadata 169](#_Toc15906)

[G.3 SMPTE-labelled metadata 169](#_Toc13625)

[Annex H 170](#_Toc21005)

[H.1 Introduction 170](#_Toc6457)

[H.2 Synchronization of RTP streams 171](#_Toc16111)

[H.3 Recording of RTP streams 172](#_Toc8078)

[H.4 Playing of recorded RTP streams 180](#_Toc32285)

[H.5 Re-sending recorded RTP streams 183](#_Toc16416)

[Annex I 184](#_Toc1081)

[I.1 Introduction 185](#_Toc28660)

[I.2 SAP properties 185](#_Toc16734)

[I.3 SAP types 185](#_Toc6466)

[Annex J 186](#_Toc20317)

[J.1 Introduction 186](#_Toc19749)

[J.2 Registration 186](#_Toc10033)

[参考文档 187](#_Toc17360)

# Foreword

ISO（国际标准化组织）和IEC（国际电工委员会）构成了世界标准化的专门体系。作为ISO或IEC成员的国家机构，通过各自组织设立的旨在处理特定技术活动领域的技术委员会，参与国际标准的制定。ISO和IEC技术委员会在共同关心的领域进行合作。其他国际组织，包括政府组织和非政府组织，与ISO和IEC保持联系，也参与了这项工作。在信息技术领域，ISO和IEC已经成立了一个联合技术委员会，ISO/IECJTC1。

国际标准按照ISO/IEC指令第2部分中规定的规则起草。

联合技术委员会的主要任务是编写国际标准。联合技术委员会通过的国际标准草案已分发给各国机构进行表决。作为国际标准出版需要至少75%的国家机构的批准。

ISO/IEC14496-12由ISO/IECJTC1联合技术委员会编写，信息技术、小组委员会SC29、音频、图片、多媒体和超媒体信息编码等部门编写。

第四版取消并取代了第三版(ISO/IEC14496-12：2008)，它构成了一个小的修订。它还包含了ISO/IEC14496-12：2008/Amd。1：2009和技术法规ISO/IEC14496-12：2008/Cor。1:2008, ISO/IEC 14496-12:2008/Cor.2:2009, ISO/IEC 14496- 12:2008/Cor.3：2009，和ISO/IEC14496-12：2008/Cor。4:2011.

ISO/IEC14496由以下各部分组成，其统称为“信息技术——视听对象的编码”：

⎯ *Part 1: Systems*

⎯ *Part 2: Visual*

⎯ *Part 3: Audio*

⎯ *Part 4: Conformance testing*

⎯ *Part 5: Reference software*

⎯ *Part 6: Delivery Multimedia Integration Framework (DMIF)*

⎯ *Part 7: Optimized reference software for coding of audio-visual objects* [Technical Report]

⎯ *Part 8: Carriage of ISO/IEC 14496 contents over IP networks*

⎯ *Part 9: Reference hardware description* [Technical Report]

⎯ *Part 10: Advanced Video Coding*

⎯ *Part 11: Scene description and application engine*

⎯ *Part 12: ISO base media file format*

⎯ *Part 13: Intellectual Property Management and Protection (IPMP) extensions*

⎯ *Part 14: MP4 file format*

⎯ *Part 15: Advanced Video Coding (AVC) file format*

⎯ *Part 16: Animation Framework eXtension (AFX)*

⎯ *Part 17: Streaming text format*

⎯ *Part 18: Font compression and streaming*

⎯ *Part 19: Synthesized texture stream*

⎯ *Part 20: Lightweight Application Scene Representation (LASeR) and Simple Aggregation Format (SAF)*

⎯ *Part 21: MPEG-J Graphics Framework eXtensions (GFX)*

⎯ *Part 22: Open Font Format*

⎯ *Part 23: Symbolic Music Representation*

⎯ *Part 24: Audio and systems interaction* [Technical Report]

⎯ *Part 25: 3D Graphics Compression Model*

⎯ *Part 26: Audio conformance*

⎯ *Part 27: 3D Graphics conformance*

⎯ *Part 28: Composite font representation*

ISO/IEC14496-12：2012的修正版本包含了ISO/IEC14496-12：2008技术勘误表5草案（未发表）所进行的修正

# Introduction

ISO基本媒体文件格式被设计为包含一个以灵活的、可扩展的格式表示的定时媒体信息，便于媒体的交换、管理、编辑和展示。此演示可以是包含该演示的系统的“本地”演示，也可以通过网络或其他流传递机制进行。

文件结构是面向对象的；一个文件可以非常简单地分解为组成对象，并直接从它们的类型推断出对象的结构。

文件格式被设计为独立于任何特定的网络协议，同时实现对它们的有效支持。

ISO基本媒体文件格式是媒体文件格式的基本格式。

预计ISO基本媒体文件格式应由WG1和WG11共同维护。因此，一个工作的分支创建了ISO/IEC15444-12和ISO/IEC14496-12，以记录ISO基础媒体文件格式，并促进联合维护。

本技术上相同的文本以MPEG-4的ISO/IEC14496-12发布，以JPEG2000的ISO/IEC15444-12形式发布，并应据此参考本规范。建议引用一个，例如ISO/IEC14496-12，并在引用中附加一个括号注释，标识另一个，例如“（技术上与ISO/IEC15444-12相同）”。

ISO和IEC对本专利权的证据、有效性和范围不采取任何立场。

该专利权的持有人已向ISO和IEC保证，他愿意以合理和非歧视性的条款和条件与世界各地的申请人协商许可证。在这方面，本专利权持有人的声明已在ISO和IEC注册。信息可从附件B所列公司获取。

需要注意的是，本文件的某些要素可能是附件B中确定的专利权的主题。ISO和IEC不负责识别任何或所有此类专利权。

ISO（www.iso.org/patents）和IEC（http://patents.iec.ch）维护与其标准相关的专利在线数据库。我们鼓励用户查阅数据库，以获取有关专利的最新信息。

**Information technology — Coding of audio-visual objects —**

**Part 12:**

**ISO base media file format**

# 1 Scope

ISO/IEC14496的这一部分规定了ISO基本媒体文件格式，这是一种通用格式，形成了许多其他更具体的文件格式的基础。这种格式包含媒体数据的定时序列的定时、结构和媒体信息，如视听presentations。

ISO/IEC14496的本部分适用于MPEG-4，但其技术内容与ISO/IEC15444-12相同，适用于JPEG2000。

# 2 Normative references

以下文件的全部或部分在本文件中正常引用，是不可缺少的。对于有日期的参考文献，只有被引用的版本适用。对于未注明日期的参考文件，适用参考文件的最新版本（包括任何修订版）。

ISO 639-2:1998, *Codes for the representation of names of languages — Part 2: Alpha-3 code*

ISO/IEC 9834-8:2005, *Information technology — Open Systems Interconnection — Procedures for the*

*operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers*

*(UUIDs) and their use as ASN.1 Object Identifier components*

ISO/IEC 11578:1996, *Information technology — Open Systems Interconnection — Remote Procedure Call*

*(RPC)*

ISO/IEC 14496-1:2010: *Information technology — Coding of audio-visual objects — Part 1: Systems*

ISO/IEC 14496-10, *Information technology — Coding of audio-visual objects — Part 10: Advanced Video*

*Coding*

ISO/IEC 14496-14, *Information technology — Coding of audio-visual objects — Part 14: MP4 file format*

ISO/IEC 15444-1, *Information technology — JPEG 2000 image coding system: Core coding system*

ISO/IEC 15444-3, *Information technology — JPEG 2000 image coding system: Motion JPEG 2000*

ISO/IEC 15938-1, *Information technology — Multimedia content description interface — Part 1: Systems*

ISO/IEC 23001-1, *Information technology — MPEG systems technologies — Part 1: Binary MPEG format for*

*XML*

ISO/IEC 23002-3, *Information technology — MPEG video technologies — Part 3: Representation of auxiliary*

*video and supplemental information*

ISO/IEC 29199-2:2012, *Information technology — JPEG XR image coding system — Part 2: Image coding*

*specification*

ISO 15076-1:2010, *Image technology colour management — Architecture, profile format and data structure —*

*Part 1: Based on ICC.1:2010*

IETF RFC 2045, *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*,

FREED, N. and BORENSTEIN, N., November 1996

IETF RFC 2046, *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*, FREED, N. and

BORENSTEIN, N., November 1996

IETF RFC 3550, *RTP: A Transport Protocol for Real-Time Applications*, SCHULZRINNE, H. et al., July 2003.

IETF RFC 3711, “*The Secure Real-time Transport Protocol (SRTP)*”, BAUGHER, M. et al., March 2004

IETF RFC 5052, *Forward Error Correction (FEC) Building Block*, WATSON, M. et al., August 2007

IETF RFC 5905, *Network Time Protocol Version 4: Protocol and Algorithms Specification*, MILLS, D., et al,

June 2010

SMIL 1.0 “*Synchronized Multimedia Integration Language (SMIL) 1.0 Specification*”,

<http://www.w3.org/TR/REC-smil/>

Rec. ITU-R TF.460-6, *Standard-frequency and time-signal emissions (Annex I for the definition of UTC.)*

# 3 Terms, definitions, and abbreviated terms

## 3.1 Terms and definitions

根据本文档的需要，采用了以下术语和定义。

### 3.1.1 box

由唯一的类型标识符和长度定义的面向对象的构建块。

注意，在一些规范中被称为“atom”，包括MP4的最初的定义。

### 3.1.2 chunk

一个track的连续样本集。

### 3.1.3 container box

其唯一目的是包含和分组一组相关box的boxes。

注：Container boxes通常不来自 “fullbox”。

### 3.1.4 hint track

不包含媒体数据，而是包含将一个或多个track打包到流媒体通道的指令。

### 3.1.5 hinter

在只包含媒体的文件上运行的工具，用于向文件中添加一个或多个hint track，从而方便流式传播（streaming）。

### 3.1.6 ISO Base Media File

符合本规范中描述的文件格式的文件的名称。

### 3.1.7 leaf subsegment

不包含任何索引信息以使其能够进一步划分为子段的子段。

### 3.1.8 media data box

可保存presentation的实际媒体数据的box（“mdat”）。

### 3.1.9 movie box

容器box，其子box定义了presentation的元数据（“moov”）。

### 3.1.10 presentation

一个或多个运动序列，可能与音频相结合。

### 3.1.11 random access point (RAP)

从附录I中定义的1或2或3型SAP的ISAU开始的track中的样本。

注：非正式的，一个样本，当解码开始时，样本本身和所有按照组合顺序排列的样本都可以被正确地解码。

### 3.1.12 random access recovery point

在显示时间等于附件一中定义的4型SAP的TSAP的跟踪中的样本。

注：非正式的样本，在以解码顺序解码之前的一些样本后，可以正确解码，有时称为逐步解码刷新。

### 3.1.13 sample

与单个时间戳关联的所有数据

注1一个track中的没有两个样本可以共享相同的时间戳。

注2在non-hint tracks中，样本例如是单个视频帧、解码顺序的一系列视频帧或解码顺序的压缩音频部分；在hint tracks

中，示例定义一个或多个流数据包的形成。

### 3.1.14 sample description

定义和描述track中一些样本的格式的结构。

### 3.1.15 sample table

打包目录，用于样本的时间和物理布局的跟踪。

### 3.1.16 sync sample

从附录I中定义的1或2型SAP的ISAU开始的track样本。

注:非正式地说，一种媒体样本，它开始一个新的独立的样本序列; 如果在sync sample处开始解码，则同步采样处和按解码顺序依次进行的采样均可被正确解码，所得到的解码样本集形成了从合成时间最早的解码样本处开始的媒体的正确呈现; 媒体格式可以为该格式提供更精确的sync sample定义。

### 3.1.17 segment

ISO基本媒体文件格式文件的一部分，包括(a)一个movie box及其关联的媒体数据（如果有）和其他相关boxes或(b)一个或多个电影片段boxes及其关联的媒体数据和其他相关boxes。

### 3.1.18 subsegment

由电影片段boxes形成的segment(片段)的时间间隔，这也是一个有效的segment(片段)。

### 3.1.19 track

相关样本的定时序列（q.V.）在一个ISO基础媒体文件中。

注：对于媒体数据，一个track对应于一序列的图像或采样的音频；对于hint tracks，一个track对应于流媒体通道。

## 3.2 Abbreviated terms

根据本文件的目的，应用了以下缩略语：

**ALC** Asynchronous Layered Coding

**FD** File Delivery

**FDT** File Delivery Table

**FEC** Forward Error Correction

**FLUTE** File Delivery over Unidirectional Transport

**IANA** Internet Assigned Numbers Authority

**LCT** Layered Coding Transport

**MBMS** Multimedia Broadcast/Multicast Service

# 4 Object-structured File Organization

## 4.1 File Structure

文件是一系列对象，在本规范中称为boxes。所有数据都包含在boxes中；文件中没有其他数据。这包括特定文件格式所需的任何初始签名。

所有符合本规范本节要求的object-structured文件（所有Object-Structured文件）都应包含一个File Type Box。

## 4.2 Object Structure

在此术语中的object是一个box。

Boxes以一个header开头，它提供了大小和类型。这个header允许紧凑或扩展尺寸（32或64位）和紧凑或扩展类型（32位或完整的通用唯一标识符，即UUID）。标准的boxes都使用紧凑型（32位），大多数boxes都使用紧凑型（32位）大小。通常只有Media Data Box(es)需要64位大小。

size字段是整个box的大小，包括size本身和type header、fields和所有包含的boxes。这便于对文件的通用解析。

boxes的定义以MPEG-4中定义的语法描述语言(SDL)给出（请参见条款2中的参考文献）。本规范中代码片段中的注释指出了提供信息的材料。

对象中的字段首先以最重要的字节放最前面存储，通常称为网络字节顺序或大端格式。当定义小于1字节或字段跨越字节边界时，位从每个字节中最重要的位分配给最不重要的位。例如，一个两位的字段后面跟着一个6位的字段其中两位的字段在这个字节中的高位。

aligned(8) class Box (unsigned int(32) boxtype,

optional unsigned int(8)[16] extended\_type) {

unsigned int(32) size;

unsigned int(32) type = boxtype;

if (size==1) {

unsigned int(64) largesize;

} else if (size==0) {

// box extends to end of file （此box 持续到文件尾）

}

if (boxtype==‘uuid’) {

unsigned int(8)[16] usertype = extended\_type;

}

}

这两个字段的语义如下：

size是一个整数，指定字节数，包括其所有字段和包含的boxes；如果大小为1，则实际大小在largesize字段；如果大小为0，则此box是文件中的最后一个，其内容持续到文件的末尾（通常仅用于Media Data Box）

type标识box类型；标准box使用紧凑类型，通常是四个可打印字符，便于识别，后续的boxes也同样如此显示。用户扩展名使用扩展类型；此时，类型字段被设置为“uuid”。

具有未被识别的类型的boxes应被忽略和跳过。

许多对象还包含一个version和flags字段：

aligned(8) class FullBox(unsigned int(32) boxtype, unsigned int(8) v, bit(24) f)

extends Box(boxtype) {

unsigned int(8) version = v;

bit(24) flags = f;

}

这两个字段的语义如下：

version是一个整数，它指定此格式的box的版本。

flags是一系列标志的映射。

## 4.3 File Type Box

### 4.3.1 Definition

Box Type: `ftyp’

Container: File

Mandatory: Yes

Quantity: Exactly one (but see below)

写入此规范版本的文件必须包含一个File Type Box。为了与此规范的早期版本兼容，文件可能与此规范一致，而不包含File Type Box。没有File Type Box的文件应该被读取，就好像它们包含一个带有Major\_brand=‘mp41’、minor\_version=0和只带一个值为 “mp41”的兼容brand的FTYP box一样。

根据本规范这部分结构的媒体文件可能与多个详细规范兼容，因此不总是可能谈论该文件的单一“type”或“brand”。这意味着文件扩展名和多用途互联网邮件扩展名(MIME)类型的效用多少减少了。

此box必须尽早放置在文件中（例如，在任何强制性签名之后，但在任何重要的可变大小的box之前，如Movie Box、Media Data Box或Free Space）。它标识哪个规范是文件的“最佳用途”，以及该规范的一个次要版本；以及该文件符合的一组其他规范。实现这种格式的读取者应该尝试读取被标记为与读取者实现的任何规范兼容的文件。因此，规范中任何不兼容的更改都应该注册一个新的“brand”标识符，以识别符合新规范的文件。

次要版本只提供信息。它不出现为兼容的brand，也不能用于确定文件与标准的一致性。它可以更精确地识别主要规格，用于检查、调试或改进的解码。

文件通常会由外部标识（例如，使用文件扩展名或mime类型），以标识“最佳用途”（主要brand），或作者认为将提供最大兼容性的brand。

本规范的这部分没有定义任何brand。但是，请参见下面的第6.3分条款，以了解符合整个规范的文件的brand，而不仅仅是本节。本规范中定义的所有文件格式brand都包含在附录E中，并总结了它们所需要的特性。

### 4.3.2 Syntax

aligned(8) class FileTypeBox

extends Box(‘ftyp’) {

unsigned int(32) major\_brand;

unsigned int(32) minor\_version;

unsigned int(32) compatible\_brands[]; // to end of the box

}

### 4.3.3 Semantics

此box标识此文件符合的规范。

每个brand都是一个可打印的四字符代码，与ISO注册，以确定一个精确的规格。

major\_brand – 是一个 brand 标识

minor\_version –是为主要brand的小版本提供信息的整数

compatible\_brands – 是一个 brands 列表，持续到 box 结尾

# 5 Design Considerations (注意事项)

## 5.1 Usage

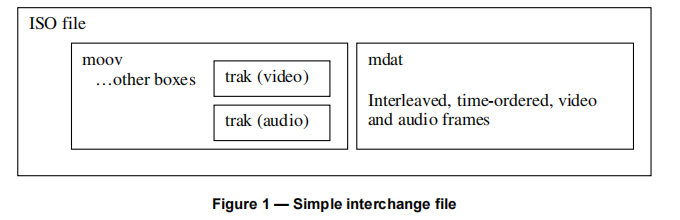
### 5.1.1 Introduction

该文件格式旨在作为许多操作的基础。在这些不同的角色中，它可以以不同的方式使用，以及在整体设计的不同方面被行使。

### 5.1.2 Interchange

当用作交换格式时，文件通常是自包含的(不引用其他文件中的媒体)，只包含表示中实际使用的媒体数据，不包含任何与流相关的信息。 这将生成一个小的、独立于协议的、自包含的文件，其中包含核心媒体数据和对其进行操作所需的信息。

下图给出了一个包含两个流的简单交换文件的示例。



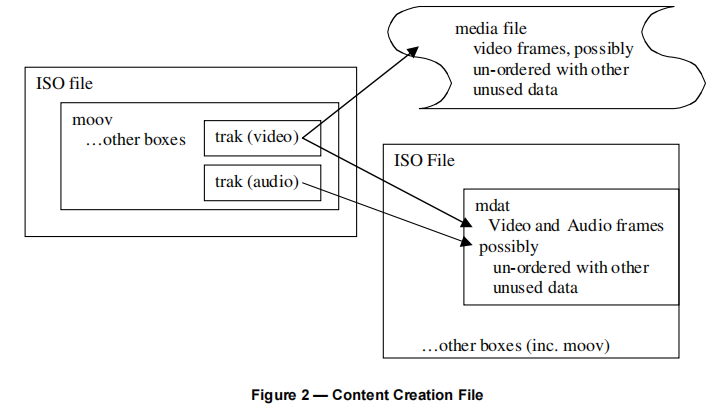
### 5.1.3 Content Creation

在内容创建过程中，可以使用该格式的一些领域，以达到有用的效果，特别是：

* 能够单独存储每个基本流（不交错），可能存储在单独的文件中。
* 在包含媒体数据和其他流的单一presentation文稿中工作的能力（例如，以未压缩的格式编辑音频track，以与已准备好的视频track对齐）。

这些特征意味着presentations可以准备、应用编辑和开发和集成内容，而不需要迭代地在光盘上重写presentation文稿——如果需要交错，必须删除未使用的数据，这将是必要的；也不需要迭代译码和重新编码数据——如果数据必须以编码状态存储，这将是必要的。

下图中显示了在内容创建过程中使用的一组文件。



### 5.1.4 Preparation for streaming

当准备流时，文件必须包含在发送信息的过程中指导流服务器的信息。此外，如果这些说明和媒体数据交织在一起，以便在提供presentation文稿时可以避免过度的寻求，这是很有帮助的。同样重要的是，必须保留原始媒体数据的完整，以便对文件进行验证、重新编辑或以其他方式重用。最后，如果可以为多个协议准备一个文件，这是很有帮助的，因此不同的服务器可能会在不同的协议上使用它。

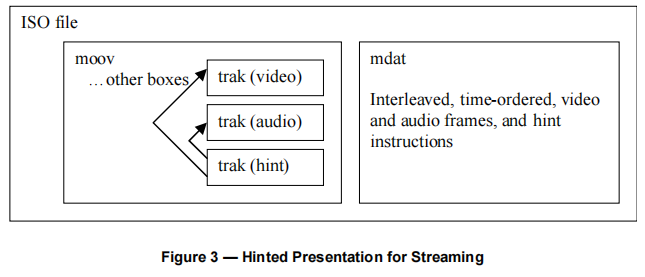
### 5.1.5 Local presentation

“本地”查看presentation文稿（即直接从文件查看，而不是通过流互连）是一个重要的应用程序；在presentation文稿分发（例如在CD或DVDROM上）、在开发过程中以及在流媒体服务器上验证内容时使用。必须支持这种本地查看，并进行完全随机访问。如果演示是在CD或DVDROM上，交错是很重要的，因为寻找可能会很慢。

### 5.1.6 Streamed presentation

当服务器从文件中进行操作以生成流时，结果流必须符合所使用的协议的规范，并且在文件本身中应该不包含文件格式信息的跟踪。服务器需要能够随机访问presentation。通过引用来自多个presentation的相同媒体数据来重用服务器内容（例如进行摘录）是有用的；如果媒体数据可以在只读媒体(例如CD)上，而在准备流媒体时不复制，仅仅是增强，它还可以帮助流媒体。

下图显示了为多路复用协议流准备的presentation，只需要一个提示跟踪。



**5.2 Design principles**

文件结构是面向对象的；文件可以非常简单地分解为组成对象，并从对象的类型直接推断出对象的结构。

媒体数据不是由文件格式“frame”的；给出媒体数据单元的大小、类型和位置的文件格式声明在物理上与媒体数据并不相邻。这使得它可以对媒体数据提取子集，并在其自然状态下使用它，而不需要复制它来为frame创建空间。元数据用于通过引用而不是包含来描述媒体数据。

类似地，特定流协议的协议信息不构成媒体数据；协议头在物理上不与媒体数据相邻。相反，媒体数据可以通过引用来包含在内。这使得在其自然状态下表示媒体数据成为可能，而不支持任何协议。它还使同一组媒体数据能够用于本地表示和多个协议。

协议信息的构建方式使得流媒体服务器只需要知道协议及其应该发送的方式；协议信息抽象了媒体的知识，因此服务器在很大程度上是与媒体类型无关的。类似地，媒体数据以一种没有协议意识的方式存储，使媒体工具与协议无关。

文件格式不要求单个表示在单个文件中。这支持对内容的子设置和重用。当与非frame方法相结合时，它还可以在未格式化为本规范的文件中包含媒体数据（例如，只包含媒体数据和没有声明性信息的“原始”文件，或已经在媒体或计算机行业中使用的文件格式）。

该文件格式是基于一组共同的设计和一组丰富的可能的结构和用法。相同的格式适用于所有的用法；不需要翻译。然而，当以特定的方式使用（例如用于本地表示）时，文件可能需要以某些方式进行结构化，以实现最佳行为（例如，数据的时间排序）。本规范没有定义规范结构规则，除非使用受限制的配置文件。

# 6 ISO Base Media File organization

## 6.1 Presentation structure

### 6.1.1 File Structure

一个presentation可能包含在几个文件中。其中一个文件包含整个presentation的元数据，并按照此规范进行了格式化。此文件也可以包含所有的媒体数据，因此该presentation是自包含的。如果使用其他文件，则无需按照本规范进行格式化；它们用于包含媒体数据，也可能包含未使用的媒体数据或其他信息。此规范仅涉及表示文件的结构。媒体数据文件的格式仅受本规范的限制，因为媒体文件中的媒体数据必须能够通过这里定义的元数据来描述。

这些其他文件可以是ISO文件、图像文件或其他格式。只有媒体数据本身，如JPEG2000图像，被存储在这些其他文件中；所有的时间和框架（位置和大小）信息都在ISO基本媒体文件中，因此辅助文件本质上是自由格式的。

如果ISO文件包含hint tracks，则引用构建提示的媒体数据的media track将保留在文件中，即使其中的数据没有被hint tracks直接引用；删除所有hint tracks后，整个未提示的演示将保留。但是，请注意，media track可能会引用有关其媒体数据的外部文件。

附件A提供了一个翔实的介绍，这可能对第一次读取者有帮助。

### 6.1.2 Object Structure

文件结构为对象序列；其中一些对象可能包含其他对象。文件中的对象序列应只包含一个presentation metadata包装器（Movie box）。它通常靠近文件的开头或结束，以便于地的位置。在此级别上找到的其他对象可能是文件类型框、空闲空间框、电影片段、元数据框或媒体数据框。

### 6.1.3 Meta Data and Media Data

metadata包含在元数据包装器（Movie box）中；媒体数据包含在同一文件、Media Data Box(es)或其他文件中。媒体数据由图像或音频数据组成；媒体数据对象或媒体数据文件可以包含其他未引用的信息。

### 6.1.4 Track Identifiers

ISO文件中使用的track标识符在该文件中是唯一的；两个track不得使用相同的标识符。

存储在Movie Header Box中的下一个track标识符值通常包含一个比文件中找到的最大track标识符值大的值。这使得在大多数情况下可以轻松地生成track标识符。但是，如果这个值等于1(32位无符号maxint)，则需要为所有添加的数据搜索未使用的track标识符。

## 6.2 Metadata Structure (Objects)

### 6.2.1 Box

将保留此处未定义的类型字段。专用扩展应通过“uuid”类型来实现。此外，在本规范的未来版本中，以下类型不会被使用，或仅在其现有意义上使用，以避免与使用本格式的早期预标准版本的现有内容发生冲突：

clip, crgn, matt, kmat, pnot, ctab, load, imap;

这些track参考类型（可在track参考框的reference\_type中找到）：

tmcd, chap, sync, scpt, ssrc.

许多boxes包含的索引值包含到其他boxes中的序列中。这些索引以值1开始（1是序列中的第一个条目）。

### 6.2.2 Data Types and fields

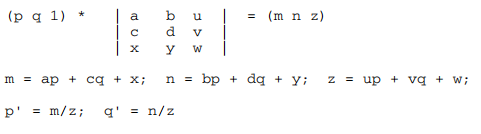
在这个规范的许多box中，有两种变体的形式：版本0使用32位字段，版本1使用相同字段使用64位大小的版本。一般来说，如果可以使用版本0 box（32位字段大小），则应该使用；版本1 box仅在需要允许的64位字段大小时使用。计数器、偏移量、时间、持续时间等的值。在此格式中，当达到其字段中可以存储的最大值时，不要“包装”为0；所有值都必须使用适当的大字段。

为了方便在内容创建过程中，文件中存储了创建和修改时间。这些数字可以是32位或64位的数字，从1904年1月1日午夜开始计算秒，这是一个方便的闰年计算日期。32位在大约2040年之前就足够了。这些时间应以世界时间坐标(UTC)表示，因此如果显示，可能需要根据当地时间进行调整。

不动点数是由一个整数除以2的幂产生的值。例如，将32位整数除以4形成30.2定点数。(PS:这里30.2的4倍是120.8，不是整数，可能这里是30.25)

在方框描述中显示为“template”的字段在使用此规范的规范中是可选的。如果在另一个规范中使用了该字段，则该使用必须与此处的定义一致，并且该规范必须定义该使用是可选的还是强制性的。类似地，在本规范的早期版本中使用了标记为“pre-defined”的字段。对于这两种字段，如果在规范中没有使用该类字段，则应该将其设置为指定的默认值。如果没有使用该字段，则必须在复制框时不检查，并在读取时忽略。

在headers中出现的矩阵值指定要表示的视频图像的转换。并非所有导出的规范都使用矩阵；如果不使用，则应将它们设置为单位矩阵。如果使用矩阵，则将点（p、q）转换为（p’、q’），如下矩阵：



坐标（p、q）在解压的帧上，坐标（p’、q’）在渲染输出处。因此，例如，矩阵{2、0、0、0、2、0、0、0、1}精确地将图像的像素维数增加了一倍。由矩阵转换的坐标不以任何方式标准化，并表示实际的样本位置。因此，例如，{x，y}可以被认为是图像的一个平移向量。

纵坐标原点位于左上角，X值向右增加，Y值向下增加。{p、q}和{p‘、q’}将分别作为相对于原始图像左上角（缩放到由track头的宽度和高度决定的大小后）和变换（渲染）表面的绝对像素位置。

每个track都是用它指定的矩阵组合成一个整体图像；然后根据MovieHeaderBox中的movie级别的矩阵进行转换和组合。例如，结果图像是否被“裁剪”，以消除窗口内的像素中没有显示的垂直矩形区域。例如，如果只显示一个视频track，并且它有一个转换为{20,30}，并且在电影标题框中有一个统一矩阵，一个应用程序可以选择不显示图像和原点之间的空的“L”形区域。

矩阵中的所有值都存储为16.16个不动点值，除了u、v和w，这些值存储为2.30个不动点值。

矩阵中的值以{a、b、u、c、d、v、x、y、w}的顺序存储。

### 6.2.3 Box Order

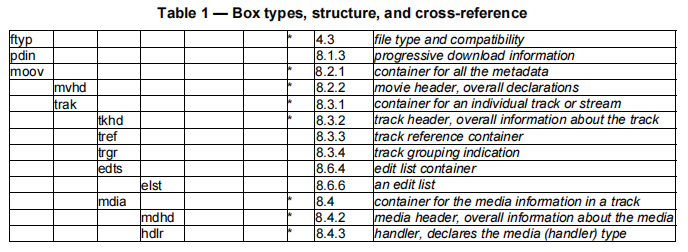
下表提供了正常封装结构的总体视图。

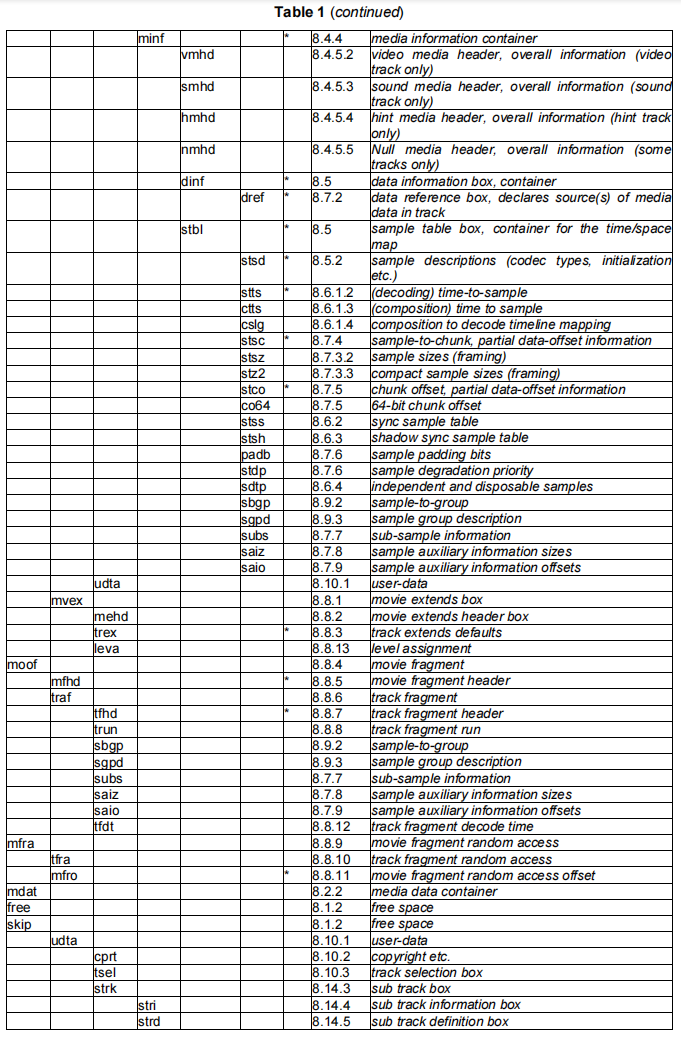
该表显示了可能出现在最左边列的顶层的那些方框；缩进用于显示可能的包含。因此，例如，曲目头盒(tkhd)可以在曲目盒(trak)中找到，它可在电影盒(moov)中找到。并非所有的框都需要在所有文件中使用；强制框用星号（\*）标记。如果没有可选框，则必须假设是什么，请参阅单个框的描述。

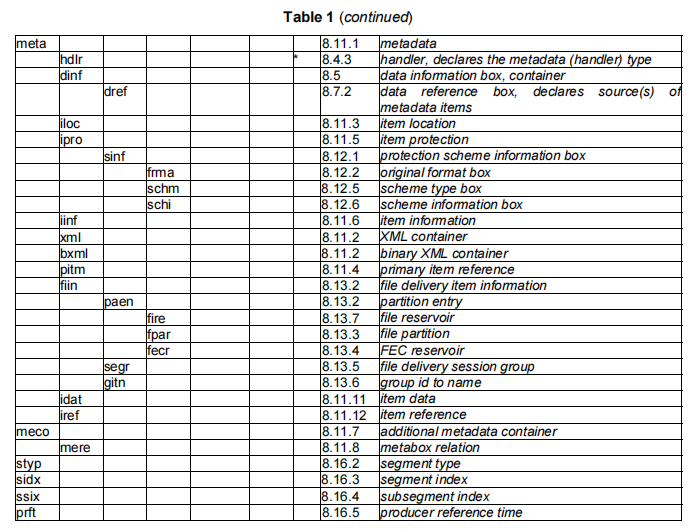
用户数据对象只能放置在Movie或track box中，使用扩展类型的对象可以放置在各种各样的容器中，而不仅仅是顶层。

为了提高文件的互操作性和实用性，box的顺序应遵循以下规则和准则：

1. 文件类型box “ftyp”应出现在任何可变长度的box（例如movie、free space、media data）之前。如果需要，在它前面只能有一个固定大小的box，如文件签名。
2. 强烈建议将所有header box首先放在它们的容器中：这些box是movie header、track header、media header以及媒体信息框中的特定movie header（例如video media header）。
3. 任何Movie Fragment Boxes都应按顺序排列（见第8.8.5款）。
4. 建议Sample Table Box中的boxes按以下顺序排列：样本说明、采样时间、采样到块、样本大小、块偏移。
5. 强烈建议Track Reference Box和编辑列表（如果有）在Media Box之前，Handler Reference Box 应该在Media Information Box之前，Data Information Box应该在Sample Table Box之前。
6. 建议将user Data Boxes放在容器中，即Movie Box 或 Track Box。
7. 如果存在Movie Fragment Random Access Box，建议它最后出现在文件中。
8. 建议将progressive download information box 尽早放在文件中，以达到最大的效用。







### 6.2.4 URIs as type indicators

当URI被用作类型指示器时（例如，在样本条目或未定时的元数据中），URI必须是绝对的，而不是相对的，并且数据的格式和意义必须由有问题的URI来定义。这种标识可以是分层的，因为URI的初始子字符串可以标识数据的总体性质或族(例如，urn：oid：标识元数据是由一个iso标准的对象标识符标识的)。

URI应该是可去引用的，但不是必需的。它可以由reader与它知道和识别的URI类型集进行字符串比较。uri为类型标识符提供了一个很大的非碰撞的非注册空间。

如果URI包含一个域名(例如，它是一个URL)，那么它也应该以mmyyyy的形式包含一个月-日期。该日期必须接近扩展定义的时间，并且URI必须是以该日期域名所有者授权的方式定义的。（这可以避免在域名更改所有权时出现的问题）。

## 6.3 Brand Identification

适用于该文件格式的品牌的定义见附录E。

# 7 Streaming Support

## 7.1 Handling of Streaming Protocols

该文件格式支持网络上的媒体数据流媒体和本地播放。发送协议数据单元的过程是基于时间的，就像基于时间的数据的显示一样，因此可以用基于时间的格式来适当地描述。一个文件或‘movie’支持流式操作，这个操作包含关于数据单元到流的信息。此信息包含在名为“hint”的文件的额外的track中。Hint tracks也可用于记录流；这些tracks被称为Reception Hint Tracks，以区别于普通（或服务器，或传输）hint tracks。

传输或服务器提示track包含帮助流媒体服务器形成传输数据包的指令。这些指令可以包含服务器要发送的即时数据（例如，报头信息）或媒体数据的参考段。这些指令在文件中编码的方式与将编辑或表示信息编码在文件中以供本地播放相同。提供的信息不是编辑或表示信息，而是允许服务器以适合于使用特定网络传输的流媒体的方式打包媒体数据的信息。

相同的媒体数据用于包含hints的文件中，无论是用于本地播放，还是通过许多不同协议的流媒体。不同协议的单独“hint”tracks可能包含在同一文件中，媒体将在所有这些协议上播放，而不会制作媒体本身的任何额外副本。此外，通过为特定的协议添加适当的hint tracks，可以很容易地使现有的媒体成为流媒体。媒体数据本身不需要以任何方式重新铸或重新格式化。

这种流媒体和记录的方法比要求将媒体信息划分为实际的传输和媒体格式传输的方法更具有空间效率。在这种方法下，本地回放要么需要从数据包中重新组装媒体，要么需要有两个媒体副本——一个用于本地回放，另一个用于流媒体。类似地，使用这种方法在多个协议上流式传输此类媒体需要每个传输的媒体数据的多个副本。这在空间上是低效的，除非媒体数据已经被大量转换为流媒体（例如，通过应用纠错编码技术，或通过加密）。

当记录一个或多个数据包流时，可以使用Reception hint tracks。Reception hint tracks指示顺序、接收时间和接收到的数据包的内容等。

注意：播放器可以根据Reception hint tracks复制接收到的包流，并将复制的包流当作新接收的数据来处理。

## 7.2 Protocol ‘hint’ tracks

对流媒体的支持是基于以下三个设计参数：

* 媒体数据被表示为一组独立于网络的标准track，可以正常播放、编辑等；
* hint tracks有一个通用的声明和基本结构；这种通用的格式与协议无关，但包含在hint tracks中描述的协议的声明；
* 可以传输的每个协议都有一个特定的hint tracks设计；所有这些设计都使用相同的基本结构。例如，可能有针对RTP（针对互联网）和MPEG-2传输（广播）的设计，或针对新标准或特定于供应商的协议的设计。

由服务器在server hint tracks的方向下发送的结果流，或从reception hint tracks中重建的结果流，不需要包含特定于文件的信息的跟踪。此设计不要求在线上数据或解码站数据中使用文件结构或声明样式。例如，一个使用ITU-TH.261视频和DVI音频的文件，通过RTP流进行传输，结果是这一路包流，完全符合IETF规范，这个规范用来将这些编码打包到RTP中。

## 7.3 Hint Track Format

Hint tracks 用于描述文件中的基本流数据。每个协议或每个相关协议家族都有自己的hint track格式。同一协议的服务器hint track格式和接收hint track格式与示例描述条目中相关的四个字符代码不同。换句话说，一个不同的四个字符的代码用于同一协议的服务器hint track和接收hint track。同一协议的服务器hint track格式和接收hint track格式的语法应该相同或兼容，这样，只要适当处理 reception hint track的潜在降级， reception hint track就可以用来重新发送流。大多数协议只需要一个示例描述格式。

服务器通过首先找到所有hint tracks，然后使用他们的协议（示例描述格式）查找服务器hint tracks。如果此时有选择，那么服务器将根据首选协议或通过比较hint tracks头中的特性或示例描述中其他特定于协议的信息进行选择。特别是在没有服务器hint tracks的情况下，服务器也可以使用其协议的reception hint track。但是，服务器应该适当地处理所使用的reception hint track所描述的接收流的潜在降级。

设置了track\_in\_movie标志的曲目是播放的候选曲目，不管它们是media tracks还是 reception hint tracks。

Hint tracks通过引用从其他tracks中拉出数据来构造流。这些其他的tracks可能是hint track或基本的stream track。这些指针的确切形式由协议的示例格式定义的，但通常它们由四条信息组成：一个track参考索引、一个示例号、一个偏移量和一个长度。其中一些可能是针对特定的协议所隐含的。这些“指针”总是指向数据的实际源。如果一个提示track是建立在另一个提示track的“顶部”，那么第二个提示track必须直接引用第一个media track使用的media track，其中来自这些media track的数据被放置在流中。

所有的hint track都使用一组公共的声明和结构。

* Hint track通过“hint”类型的track引用链接到它们所携带的基本流track
* 它们在Handler Reference Box中使用了一种处理程序类型的“hint”
* 它们使用一个 Hint Media Header Box
* 它们使用一个样本描述里的 hint sample entry，具有它们所表示的协议唯一的名称和格式

服务器hint tracks通常被标记为禁用的本地播放，与他们的track头track\_in\_movie和track\_in\_preview标志设置为0。

hint tracks可以由创作工具创建，也可以通过提示工具添加到现有的presentation中。这样的工具是媒体和协议之间的“桥梁”，因为它非常理解两者。这允许创作工具理解媒体格式，但不能了解协议，并允许服务器理解协议（及其hint tracks），但不能了解媒体数据的细节。

hint tracks不使用单独的合成时间；“ctts”表不出现在hint tracks中。提示的过程会正确地计算出传输时间作为解码时间。

注1：使用reception hint tracks作为发送接收流的hint的服务器应处理接收流的潜在降级，例如传输延迟抖动和包丢失，优雅地确保遵守协议和包含数据格式的约束，而不管接收流的潜在降级。

注2：将接收的流转换为media track，使符合ISO基本媒体文件格式早期版本的现有播放器能够处理录制的文件，只要该媒体格式得到支持。 然而，大多数媒体编码标准只规定对无错误流的解码，因此应该确保media track中的内容能够被正确解码。播放者可以利用reception hint tracks 来处理由传输引起的降级，也就是说，可能无法正确解码的内容只位于reception hint tracks 中。 通过将来自media track的数据引用到reception hint tracks 中，可以避免在media track和reception hint tracks 中都有正确媒体样本的副本的需要。

# 8 Box Structures

## 8.1 File Structure and general boxes

### 8.1.1 Media Data Box

#### 8.1.1.1 Definition

Box Type: ‘mdat’

Container: File

Mandatory(强制): No

Quantity(数量): Zero or more

此box中包含媒体数据。在video track中，这个box将包含视频帧。一个presentation可能包含零个或多个media data box。实际的媒体数据遵循类型字段；其结构由元数据描述（请参见示例表子条款8.5和item location box子条款8.11.3）。

在大型presentation中，可能希望在这个box中有比32位大小允许的更多的数据。在这种情况下，使用了上述子条款4.2中所述的大小字段的大变体。

文件中可能有任意数量的这些box（如果所有媒体数据都在其他文件中，则包括零）。元数据指文件中的绝对偏移量（见子条款8.7.5，块偏移框），因此很容易跳过media data box headers和可用空间，也可以引用和使用没有任何box结构的文件。

#### 8.1.1.2 Syntax

aligned(8) class MediaDataBox extends Box(‘mdat’) {

bit(8) data[];

}

#### 8.1.1.3 Semantics

data是包含的 media data

### 8.1.2 Free Space Box

#### 8.1.2.1 Definition

Box Types: ‘free’, ‘skip’

Container: File or other box

Mandatory: No

Quantity: Zero or more

free-space box的内容不相关，可以忽略或删除对象，而不影响presentation。（在删除对象时应小心，因为这可能会使示例表中使用的偏移量无效，除非此对象位于所有媒体数据之后）。

#### 8.1.2.2 Syntax

aligned(8) class FreeSpaceBox extends Box(free\_type) {

unsigned int(8) data[];

}

#### 8.1.2.3 Semantics

free\_type 可能是 ‘free’ 或 ‘skip’.

### 8.1.3 Progressive Download Information Box

#### 8.1.3.1 Definition

Box Types: ‘pdin’

Container: File

Mandatory: No

Quantity: Zero or One

Progressive download information box有助于渐进下载ISO文件。该box包含成对的数字（到box的末尾），指定以字节/秒为单位的有效文件下载比特率和以毫秒为单位的建议初始播放延迟的组合。

接收方可以估计它所经历的下载速率，并通过对之间的线性插值或从第一个或最后一个条目外推获得合适的初始延迟的上限估计。

建议将渐进式下载信息box尽早放在文件中，以获得最大的效用。

#### 8.1.3.2 Syntax

aligned(8) class ProgressiveDownloadInfoBox

extends FullBox(‘pdin’, version = 0, 0) {

for (i=0; ; i++) { // to end of box

unsigned int(32) rate;

unsigned int(32) initial\_delay;

}

}

**8.1.3.3 Semantics**

rate是一个以字节/秒表示的下载速率

initial\_delay是在播放文件时建议使用的延迟，这样，如果下载以给定的速度继续，文件中的所有数据将及时到以供使用，播放不需要停止。

## 8.2 Movie Structure

### 8.2.1 Movie Box

#### 8.2.1.1 Definition

Box Type: ‘moov’

Container: File

Mandatory: Yes

Quantity: Exactly one

presentation的元数据存储在文件顶层的单个movie box中。通常这个box靠近文件的开头或结尾，尽管这不是必需的。

8.2.1.2 Syntax

aligned(8) class MovieBox extends Box(‘moov’){

}

### 8.2.2 Movie Header Box

#### 8.2.2.1 Definition

Box Type: ‘mvhd’

Container: Movie Box (‘moov’)

Mandatory: Yes

Quantity: Exactly one

这个box定义了独立于媒体的整体信息，并且与整个presentation的整体相关。

#### 8.2.2.2 Syntax

aligned(8) class MovieHeaderBox extends FullBox(‘mvhd’, version, 0) {

if (version==1) {

unsigned int(64) creation\_time;

unsigned int(64) modification\_time;

unsigned int(32) timescale;

unsigned int(64) duration;

} else { // version==0

unsigned int(32) creation\_time;

unsigned int(32) modification\_time;

unsigned int(32) timescale;

unsigned int(32) duration;

}

template int(32) rate = 0x00010000; // typically 1.0

template int(16) volume = 0x0100; // typically, full volume

const bit(16) reserved = 0;

const unsigned int(32)[2] reserved = 0;

template int(32)[9] matrix =

{ 0x00010000,0,0,0,0x00010000,0,0,0,0x40000000 };

// Unity matrix

bit(32)[6] pre\_defined = 0;

unsigned int(32) next\_track\_ID;

}

#### 8.2.2.3 Semantics

version是一个整数，它指定此box的版本（本规范中的0或1）

creation\_time是一个整数，它声明了presentation的创建时间(自1904年1月1日午夜开始，以UTC时间为单位)

modification\_time是一个整数，它声明presentation被修改的最近一次时间(从1904年1月1日午夜开始，以UTC时间计秒)

timescale是一个整数，它指定了整个表示形式的时间尺度；这是在1秒内传递的时间单位数。例如，以6秒为单位的时间坐标系统的时间尺度为60。

duration是一个整数，它声明表示形式的长度（在指定的时间尺度中）。此属性来源于presentation的box：该字段的值对应于presentation中最长的track的持续时间。如果不能确定持续时间，则将持续时间设置为所有1s。

rate是一个固定点16.16的数字，表示播放presentation的首选速率；1.0(0x00010000)是正常的向前播放

volume是一个固定点8.8数字，表示首选的播放音量。1.0（0x0100）为全容量。

matrix为视频提供了一个变换矩阵；(u、v、w)在这里被限制为（0,0,1），十六进制值(0、0、0x40000000)。

next\_track\_ID是一个非零整数，它表示要添加到此presentation中的下一个track的track ID要使用的值。零不是一个有效的跟踪ID值。next\_track\_ID的值应大于使用中的最大trackID。如果此值等于所有1s(32位maxint)，并且要添加一个新的媒体track，则必须在文件中搜索未使用的track标识符。

## 8.3 Track Structure

### 8.3.1 Track Box

#### 8.3.1.1 Definition

Box Type: ‘trak’

Container: Movie Box (‘moov’)

Mandatory: Yes

Quantity: One or more

这是一个用于单track presentation的容器box。一个presentation由一个或多个track组成。每条track都独立于其他track，并携带着自己的时间和空间信息。每个track都将包含其相关联的media box。

track用于两个目的：(a)用于包含媒体数据（media track），(b)用于包含流媒体协议的分组信息（hint track）

ISO文件中应至少有一个media track，所有导致hint track的media track应保留在文件中，即使其中的媒体数据未被hint track引用；删除所有hint track后，应保留整个未提示的呈现。

#### 8.3.1.2 Syntax

aligned(8) class TrackBox extends Box(‘trak’) {

}

### 8.3.2 Track Header Box

#### 8.3.2.1 Definition

Box Type: ‘tkhd’

Container: Track Box (‘trak’)

Mandatory: Yes

Quantity: Exactly one

此box指定单track的特征。在一个track中只包含一个Track Header Box。

在没有编辑列表的情况下，track的presentation从整个表示的开始部分开始。一个空的编辑被用于偏移一个track的开始时间。

Media track的track header标志的默认值为7(track\_enabled、track\_in\_movie、track\_in\_preview)。如果在presentation中，所有track都没有设置track\_in\_movie或track\_in\_preview，那么所有track将被视为两个标志都设置在所有track上。服务器提示track的track\_in\_movie和track\_in\_preview应该设置为0，以便在本地播放和预览时忽略它们。

在“iso3” brand或具有相同要求的brand下，track header的宽度和高度是在一个概念上的“正方形”（统一的）网格上测量的。在由叠加或合成系统引起的任何转换或放置之前，跟踪视频数据被归一化为这些维度（逻辑上）。如果使用track（和电影）矩阵，也可以在这个均匀比例的空间中操作。

#### 8.3.2.2 Syntax

aligned(8) class TrackHeaderBox

extends FullBox(‘tkhd’, version, flags){

if (version==1) {

unsigned int(64) creation\_time;

unsigned int(64) modification\_time;

unsigned int(32) track\_ID;

const unsigned int(32) reserved = 0;

unsigned int(64) duration;

} else { // version==0

unsigned int(32) creation\_time;

unsigned int(32) modification\_time;

unsigned int(32) track\_ID;

const unsigned int(32) reserved = 0;

unsigned int(32) duration;

}

const unsigned int(32)[2] reserved = 0;

template int(16) layer = 0;

template int(16) alternate\_group = 0;

template int(16) volume = {if track\_is\_audio 0x0100 else 0};

const unsigned int(16) reserved = 0;

template int(32)[9] matrix=

{ 0x00010000,0,0,0,0x00010000,0,0,0,0x40000000 };

// unity matrix

unsigned int(32) width;

unsigned int(32) height;

}

#### 8.3.2.3 Semantics

version是一个整数，它指定此box的版本（本规范中的0或1）

flag是带有标志的24位整数；定义了以下值：Track\_enabled：表示track处于启用状态。标志值为0x000001。禁用的track（低位为零）被视为不存在。Track\_in\_movie：表示在presentation中使用了该track。标志值为0x000002。Track\_in\_preview：表示在预览presentation时使用了该track。标志值为0x000004。

creation\_time是一个整数，它声明该track的创建时间(自1904年1月1日午夜开始，以UTC时间为单位)

modification\_time是一个整数，它声明track最近被修改的时间(从1904年1月1日午夜开始，以UTC时间计秒)

track\_ID是一个整数，它在此presentation的整个生命周期内唯一地标识此track。跟踪id永远不会被重用，也不能为零。

duration是一个整数，表示此track的持续时间（在movie header box中指示的时间尺度内）。该字段的值等于所有track编辑的持续时间之和。如果没有编辑列表，则持续时间是样本持续时间的总和，并被转换为movie header box中的时间尺度。如果不能确定该track的持续时间，则将持续时间设置为所有1s。

layer指定视频track的前后顺序；数字较低的track更接近查看器。0是正常值，-1将在track0的前面，以此类推。

alternate\_group是一个指定一组或多个track集合的整数。如果此字段为0，则没有关于与其他track的可能关系的信息。如果此字段不是0，则对于彼此包含备用数据的track应该相同，而对于属于不同这些组的track则不同。在一个替代组中只有一个track应该在任何时间播放或流，并且必须通过比特率、编解码器、语言、包大小等属性与组中的其他track区分开来。一个组可能只有一个成员。

volume是一个固定的8.8值，指定track的相对音频音量。全体积为1.0（0x0100），为正常值。它的价值对于纯粹的视觉track无关。曲目可以根据音量组合它们，然后使用整个movie header box音量设置；或者可以使用更复杂的音频组合（例如MPEG-4BIFS）。

matrix为视频提供了一个变换矩阵；(u、v、w)在这里被限制为（0,0,1）、十六进制(0、0、0x40000000)。

width和height将track的视觉显示大小指定为固定点16.16个值。这些不需要与图像的像素尺寸相同，这是记录在样本描述(s)中；在对由矩阵所表示的track进行任何整体变换之前，序列中的所有图像都被缩放到这个大小。图像的像素尺寸是默认值。

### 8.3.3 Track Reference Box

8.3.3.1 Definition

Box Type: `tref’

Container: Track Box (‘trak’)

Mandatory: No

Quantity: Zero or one

这个box提供了一个从包含的track到presentation中的另一个track的引用。键入这些引用。一个“hint”引用链接，从包含的提示track链接到它所提示的媒体数据。内容描述引用“cdsc”将描述性或元数据跟踪链接到它所描述的内容。“hind”依赖关系表示被引用的track可能包含对包含track引用的track的解码所需的媒体数据。参考track应为提示track。例如，“hind”依赖关系可以用于指示通过RTP记录分层IP多播的hint tracks之间的依赖关系。

track box中完全可以包含一个track参考框。

如果此box不存在，则该track不会以任何方式引用任何其他track。引用数组的大小可以填充引用类型box。

#### 8.3.3.2 Syntax

aligned(8) class TrackReferenceBox extends Box(‘tref’) {

}

aligned(8) class TrackReferenceTypeBox (unsigned int(32) reference\_type) extends

Box(reference\_type) {

unsigned int(32) track\_IDs[];

}

#### 8.3.3.3 Semantics

track reference box中包含track参考类型box。

track\_ID是一个整数，它提供了从包含的track到presentation中的另一个track的引用。track\_IDs从不被重复使用，也不能等于零。

reference\_type应设置为以下值之一，或注册的值或来自派生规范或注册的值：

‘hint’ 被引用的track(s)包含此hint track的原始媒体

‘cdsc’ 这个track描述了被引用的track。

“hind” 这个track依赖于引用的hint track，也就是说，只有在使用引用的hint track时，才应该使用它。

“vdep” 这track包含了参考视频track的辅助深度视频信息

“vplx” 此track包含参考视频track的辅助视差视频信息

### 8.3.4 Track Group Box

#### 8.3.4.1 Definition

Box Type: ‘trgr’

Container: Track Box (‘trak’)

Mandatory: No

Quantity: Zero or one

此box允许指示track组，其中每个组共享一个特定的特征，或者组内的track具有一个特定的关系。box包含零个或多个box，特定特征或关系由所包含box的box类型表示。包含的box包含一个标识符，可用于总结属于同一track组的track。在track组box中包含相同类型的包含box，并且在这些包含box中具有相同的标识符值的track属于同一track组。

track组不得用于表示track之间的依赖关系。相反，track参考box被用于这些目的。

8.3.4.2 Syntax

aligned(8) class TrackGroupBox('trgr') {

}

aligned(8) class TrackGroupTypeBox(unsigned int(32) track\_group\_type) extends

FullBox(track\_group\_type, version = 0, flags = 0)

{

unsigned int(32) track\_group\_id;

// the remaining data may be specified for a particular track\_group\_type

}

#### 8.3.4.3 Semantics

track\_group\_type表示分组类型，并应设置为以下值之一，或已注册的值，或来自派生的规范或注册的值：

* “msrc” 表示此track属于多源表示形式。在track\_group\_type ‘msrc’的组类型box中具有相同的track\_group\_id值的track被映射为来自相同的源。例如，视频电话呼叫的记录可能对两个参与者都有音频和视频，并且与一个参与者的音频track和视频track相关联的track\_group\_id值与与另一个参与者的track相关联的track\_group\_id值不同。

track\_group\_id和track\_group\_type对标识了文件中的一个跟踪组。包含具有相同track\_group\_id值的特定track组类型box的track属于同一track组。

## 8.4 Track Media Structure

### 8.4.1 Media Box

#### 8.4.1.1 Definition

Box Type: ‘mdia’

Container: Track Box (‘trak’)

Mandatory: Yes

Quantity: Exactly one

媒体声明容器包含在track中声明有关媒体数据的信息的所有对象。

#### 8.4.1.2 Syntax

aligned(8) class MediaBox extends Box(‘mdia’) {

}

**8.4.2 Media Header Box**

Box Type: ‘mdhd’

Container: Media Box (‘mdia’)

Mandatory: Yes

Quantity: Exactly one

媒体标题声明的总体信息与媒体无关，并且与track中的媒体特征相关。

#### 8.4.2.2 Syntax

aligned(8) class MediaHeaderBox extends FullBox(‘mdhd’, version, 0) {

if (version==1) {

unsigned int(64) creation\_time;

unsigned int(64) modification\_time;

unsigned int(32) timescale;

unsigned int(64) duration;

} else { // version==0

unsigned int(32) creation\_time;

unsigned int(32) modification\_time;

unsigned int(32) timescale;

unsigned int(32) duration;

}

bit(1) pad = 0;

unsigned int(5)[3] language; // ISO-639-2/T language code

unsigned int(16) pre\_defined = 0;

}

#### 8.4.2.3 Semantics

version是一个整数，它指定此box的版本（0或1）

creation\_time是一个整数，它声明此track中媒体的创建时间(从1904年1月1日午夜开始，以UTC时间计)

modification\_time是一个整数，它声明该track中的媒体最近被修改的时间（自1904年1月1日午夜以来，以秒为单位）

timescale是指定的整数，指定此介质的时间尺度；这是一秒内传递的时间单位数。例如，以6秒为单位的时间坐标系统的时间尺度为60。

duration是一个整数，它声明此介质的持续时间（在时间尺度的尺度中）。如果不能确定持续时间，则将持续时间设置为所有1s。

language声明了此媒体的语言代码。有关三个字符代码集，请参见ISO639-2/T。每个字符都被打包为其ASCII值与0x60之间的差值。由于代码仅限于三个小写字母，因此这些值是严格的正值。

### 8.4.3 Handler Reference Box

8.4.3.1 Definition

Box Type: ‘hdlr’

Container: Media Box (‘mdia’) or Meta Box (‘meta’)

Mandatory: Yes

Quantity: Exactly one

媒体box中的这个框声明了track中的媒体数据呈现的过程，从而声明了track中的媒体的性质。例如，一个视频track将由一个视频处理程序来处理。

当此box出现在meta box中时，将声明“meta”box内容的结构或格式。

对于任何类型的元数据流，都有一个通用的处理程序；特定的格式由示例条目标识，例如，对于视频或音频。如果它们是文本格式，则提供一种MIME格式来记录其格式；如果在XML中，则每个示例都是一个完整的XML文档，并且还提供了XML的名称空间。

辅助视频track的编码与视频track相同，但使用这种不同的处理程序类型，并不打算进行视觉显示（例如，它包含深度信息，或其他单色或颜色二维信息）。辅助视频track通常通过一个适当的track参考链接到一个视频track。

注意，MPEG-7流是一种特定类型的元数据流，它有自己的处理程序声明，并以MP4文件格式记录下来[ISO/IEC14496-14]。

注意，元数据track使用“cdsc”类型的track-reference链接到它们描述的track。元数据track使用一个空的媒体头(“nmhd”)，如子条款8.4.5.5中定义的。

#### 8.4.3.2 Syntax

aligned(8) class HandlerBox extends FullBox(‘hdlr’, version = 0, 0) {

unsigned int(32) pre\_defined = 0;

unsigned int(32) handler\_type;

const unsigned int(32)[3] reserved = 0;

string name;

}

#### 8.4.3.3 Semantics

version是一个指定此box的版本的整数

handler\_type出现在媒体box中，是包含以下值之一的整数，或来自派生规范的值：

‘vide’ Video track

‘soun’ Audio track

‘hint’ Hint track

‘meta’ Timed Metadata track

‘auxv’ Auxiliary Video track

handler\_type 当出现在元box中时，它包含一个适当的值来表示元box内容的格式。值“null”可以在主元box中使用，以表示它仅被用于保存资源。

name是UTF-8te中的一个以空结尾的字符串，它为track类型提供了一个人类可读的名称（用于调试和检查的目的）。

### 8.4.4 Media Information Box

#### 8.4.4.1 Definition

Box Type: ‘minf’

Container: Media Box (‘mdia’)

Mandatory: Yes

Quantity: Exactly one

此box包含在track中声明介质的特征信息的所有对象。

#### 8.4.4.2 Syntax

aligned(8) class MediaInformationBox extends Box(‘minf’) {

}

### 8.4.5 Media Information Header Boxes

#### 8.4.5.1 Definition

Box Types: ‘vmhd’, ‘smhd’, ’hmhd’, ‘nmhd’

Container: Media Information Box (‘minf’)

Mandatory: Yes

Quantity: Exactly one specific media header shall be present

每个track类型都有不同的媒体信息头（对应于媒体处理程序类型）；匹配的头可以是这里定义的头之一，或在派生规范中定义的头。

#### 8.4.5.2 Video Media Header Box

视频媒体头包含视频媒体独立于编码的一般表示信息。请注意，标志字段的值为1。

##### 8.4.5.2.1 Syntax

aligned(8) class VideoMediaHeaderBox

extends FullBox(‘vmhd’, version = 0, 1) {

template unsigned int(16) graphicsmode = 0; // copy, see below

template unsigned int(16)[3] opcolor = {0, 0, 0};

}

8.4.5.2.2 Semantics

version是一个指定此box的版本的整数

图形模式指定了这个视频trackc的组合模式，从以下枚举的集合，可以通过派生的规范进行扩展：复制=0复制到现有的图像

Opcolor是一组3种颜色的值（红色、绿色、蓝色），可供图形模式使用

#### 8.4.5.3 Sound Media Header Box

声音媒体报头包含用于音频媒体的独立于编码的一般表示信息。这个标题用于所有包含音频的track。

##### 8.4.5.3.1 Syntax

aligned(8) class SoundMediaHeaderBox

extends FullBox(‘smhd’, version = 0, 0) {

template int(16) balance = 0;

const unsigned int(16) reserved = 0;

}

##### 8.4.5.3.2 Semantics

version是一个指定此box的版本的整数

balance是一个定点8.8数字，将单声道放在立体声空间中；0为中心（正常值）；左为-1.0，右为1.0。

#### 8.4.5.4 Hint Media Header Box

Hint media header包含独立于协议的hint track的一般信息。（PDU是一个协议数据单元。）

##### 8.4.5.4.1 Syntax

aligned(8) class HintMediaHeaderBox

extends FullBox(‘hmhd’, version = 0, 0) {

unsigned int(16) maxPDUsize;

unsigned int(16) avgPDUsize;

unsigned int(32) maxbitrate;

unsigned int(32) avgbitrate;

unsigned int(32) reserved = 0;

}

##### 8.4.5.4.2 Semantics

version是一个指定此box的版本的整数

MaxPDUsize表示此（提示）流中最大PDU的字节大小

avgPDUsize给出了整个presentation中PDU的平均大小

maxbitrate给出了在1秒的任何窗口中以比特/秒为单位的最大速率

avgbitrate给出了在整个presentation过程中以比特/秒为单位的平均速率

#### 8.4.5.5 Null Media Header Box

除视觉和音频以外的流（例如，定时元数据流）可以使用空Media Header Box，比如这里的定义。

##### 8.4.5.5.1 Syntax

aligned(8) class NullMediaHeaderBox

extends FullBox(’nmhd’, version = 0, flags) {

}

##### 8.4.5.5.2 Semantics

version是一个指定此box的版本的整数

flags-是一个带有标志的24位整数（目前均为零）。

## 8.5 Sample Tables

8.5.1 Sample Table Box

#### 8.5.1.1 Definition

Box Type: ‘stbl’

Container: Media Information Box (‘minf’)

Mandatory: Yes

Quantity: Exactly one

sample table包含了一个track中的媒体samples的所有时间和数据索引。使用这里的表格，可以及时定位样本，确定其类型（例如是否使用I帧），并确定它们的大小、容器和在该容器中的偏移量。

如果包含“Sample Table Box”的track没有引用任何数据，则Sample Table Box不需要包含任何子box（这不是非常有用的媒体跟踪）。

如果Sample Table Box中包含的track是参考数据，则需要以下子box：Sample Description, Sample Size, Sample To Chunk, and Chunk Offset。此外，Sample Description box应至少包含一个条目。需要一个Sample Description Box，因为它包含数据引用索引字段，该字段指示要使用哪个Data Reference Box 来检索媒体示例。如果没有Sample Description，就不可能确定介质样本存储的位置。Sync Sample Box是可选的。如果不存在Sync Sample Box，则所有样本都是同步样本。

A.7提供了对使用Sample Table Box中定义的结构的随机访问的叙述性描述。

#### 8.5.1.2 Syntax

aligned(8) class SampleTableBox extends Box(‘stbl’) {

}

### 8.5.2 Sample Description Box

#### 8.5.2.1 Definition

Box Types: ‘stsd’

Container: Sample Table Box (‘stbl’)

Mandatory: Yes

Quantity: Exactly one

sample description table提供了关于所使用的编码类型的详细信息，以及该编码所需的任何初始化信息。

在输入计数之后存储在sample description box中的信息既是这里记录的track-type，也可以在track type包含变体（例如，不同的编码可能在某些公共字段之后使用不同的特定信息，甚至在视频track中）。

对于视频track，使用可视化图像条目，对于音频track，对于元数据track，使用元数据图像条目。提示跟踪使用特定于其协议的条目格式，并具有适当的名称。

对于hint tracks，sample description包含了正在使用的流协议的适当声明性数据，以及hint track的格式。sample description的定义是特定于协议的。

可以在一个track中使用多个描述。

“protocol”和“codingname”字段是注册标识符，用来唯一标识要使用的流媒体协议或压缩格式解码器。给定的协议或编码名可以对示例描述具有可选的或必需的扩展（例如，编解码器初始化参数）。所有这些扩展都应在box内；这些box出现在所要求的字段之后。未被识别的box应被忽略。

如果样本输入的“格式”字段未被识别，则不得解码样本描述本身或相关的媒体样本。

注意：示例条目的定义按特定的顺序指定了box，在派生规范中通常也遵循这一点。为了最大限度地兼容，编写器应该按照规范中的顺序和继承中隐含的顺序来构造文件，而阅读器应该准备接受任何box顺序。

采样率、采样大小和频道计数字段记录了此媒体的默认音频输出播放格式。一个audio track的时间尺度应该被选择来匹配采样率，或者是采样率的整数倍，以实现更精确的采样计时。频道计数是一个大于零的值，它表示音频可以提供的最大频道数。频道计数为1表示单声道音频，2表示立体声音响（左/右）。当使用大于2的值时，编解码器配置应识别通道分配。

在video track中，frame\_count字段必须为1，除非媒体格式的规范明确地记录了此模板字段并允许更大的值。该规范必须记录如何找到各个视频帧（它们的大小信息）和它们的时间。这个时间可能就像将样本持续时间除以帧计数来建立帧持续时间一样简单。

注意，尽管计数是32位，但项目的数量通常要少得多，并且受到样本表中的引用索引只有16位的限制

在任何元数据采样项的末尾都可以出现一个可选的比特率box，以表示流的比特率信息。这可以用于缓冲区配置。对于XML元数据，可以使用它来选择适当的内存表示格式(DOM，STX)。

视频示例输入文档中的宽度和高度是编解码器将提供的像素计数；这就允许分配缓冲区。由于这些都是计数，所以它们不考虑像素高宽比。

视频的像素长宽比和清洁光径可以分别使用“pasp”和“clap” sample entry boxes指定。这两种都是可选的；如果存在，它们会在特定于视频编解码器的结构中覆盖声明（如果有的话），如果没有这些box，就应该检查哪些结构。为了最大限度地提高兼容性，这些box应该遵循派生规范中定义或要求的任何box。

在PixelAspectRatioBox中，空间和间距有相同的单位，但这些单位是不指定的：只有比例重要。h间距和v间距可能减少，也可能不减少，它们可以减少到1/1。这两者都必须是正数。

它们被定义为一个像素的长宽比，以任意单位表示。如果一个像素出现H宽和V高，那么hSpacing/vSpacing等于H/V。这意味着显示器上n个像素高的正方形需要n个像素宽才能显示为正方形。

注意：在调整像素长宽比时，如果需要，视频的水平尺寸通常会被缩放（即，如果最终显示系统与视频源具有不同的像素长宽比)。

注意：建议尽可能通过管道进行原始像素和合成变换。如果将“校正”像素长宽比转换为正方形网格、归一化为track尺寸、合成或位置（例如track和/或movie矩阵），并归一化为显示特征，是一个单位矩阵，则不需要进行重新采样。特别是，如果可能的话，在渲染过程中，视频不应被重新采样多次。

在CleanApertureBox中，概念上有四个值。这些参数用一个分数表示。这个分数可以是也可以不是简化项。我们将这些参数对fooN和fooD称为foo。对于horizOff和vertOff，D必须为正，N可能是正或负。对于cleanApertureWidth 和 cleanApertureHeight，N和D都必须是正的。

注意：这些都是分数，原因有几个原因。首先，在某些系统中，像素高宽比校正后的精确宽度是积分的，而不是该校正前的像素计数。其次，如果视频在全光圈中调整大小，那么干净光圈的确切表达式可能不是完整的。最后，因为这是用中心和偏移量来表示的，所以需要除以2，因此可以出现半值。

考虑像素尺寸定义的视觉样本入口宽度和高度。如果图像的图像中心位于pcX和pcY，则水平度和垂直度定义如下：

pcX = horizOff + (width - 1)/2

pcY = vertOff + (height - 1)/2;

通常，horizOff和vertOff为零，因此图像围绕图像中心。

清洁光圈中最左边/最右边的像素和最上面/最底部的线位于：

pcX ± (cleanApertureWidth - 1)/2

pcY ± (cleanApertureHeight - 1)/2;

样本条目中的音频输出格式（采样大小、采样大小和信道计数字段）应该仅在不记录其自己的输出配置的编解码器时才被认为是确定的。如果音频编解码器具有关于输出格式的明确信息，则视为确定信息；在这种情况下，可以忽略样本输入中的采样量、采样大小和通道计数字段，尽管应选择合理的值（例如，可能的最高采样率）

在一个box中，该入口项包含定义元数据形式的URI和可选的初始化数据。示例和初始化数据的格式均由URI表单的全部或部分定义。

像往常一样，可以在urimetasample条目中使用一个可选的比特率box。

URI可能识别了一种元数据格式，该格式允许在每个示例中有多个“stated fact”。然而，在这种格式下的所有元数据示例实际上都是“I帧”，为它们所覆盖的时间间隔定义了整个元数据集。这意味着在任何时刻，对于给定的track，完整的元数据集包含在(a)描述该track的按时间排列的样本(如果有)，加上(b)track元数据(如果有)，movie元数据(如果有)和文件元数据(如果有)中。

如果需要逐步更改的元数据，则MPEG-7框架将提供该功能。

关于一些元数据系统的URI表单的信息见附录G。

颜色信息可以放在一个视觉样本输入中的一个或多个ColourInformationBoxes中提供。这些数据应该在样本条目中依次排列，从最准确的（也可能是最难处理的）开始，从进展到最少。这些都是咨询和关注的渲染和颜色转换，并没有与之相关的规范行为；读取者可以选择使用最合适的方法。可以忽略颜色类型未知的ColourInformationBox。

如果使用，一个ICC配置文件可能是一个受限制的配置文件，在代码“rICC”下，这允许更简单的处理。该配置文件应属于ISO15076-1所定义的基于单色或基于三成分矩阵的输入配置文件类别。如果配置文件是另一个类，则必须使用“prof”指示器。

如果在此box和视频位流中都提供了颜色信息，则此box具有优先级，并覆盖位流中的信息。

注意：当指定了ICC配置文件时，如果需要为ICC配置文件所描述的颜色主矩阵形成Y‘CbCr到R’G‘B’转换矩阵，SMPTERP177“基本电视颜色方程的推导”可能会有帮助。

#### 8.5.2.2 Syntax

aligned(8) abstract class SampleEntry (unsigned int(32) format)

extends Box(format){

const unsigned int(8)[6] reserved = 0;

unsigned int(16) data\_reference\_index;

}

class HintSampleEntry() extends SampleEntry (protocol) {

unsigned int(8) data [];

}

class BitRateBox extends Box(‘btrt’){

unsigned int(32) bufferSizeDB;

unsigned int(32) maxBitrate;

unsigned int(32) avgBitrate;

}

class MetaDataSampleEntry(codingname) extends SampleEntry (codingname) {

}

class XMLMetaDataSampleEntry() extends MetaDataSampleEntry (’metx‘) {

string content\_encoding; // optional

string namespace;

string schema\_location; // optional

BitRateBox (); // optional

}

class TextMetaDataSampleEntry() extends MetaDataSampleEntry (‘mett’) {

string content\_encoding; // optional

string mime\_format;

BitRateBox (); // optional

}

aligned(8) class URIBox

extends FullBox(‘uri ’, version = 0, 0) {

string theURI;

}

aligned(8) class URIInitBox

extends FullBox(‘uriI’, version = 0, 0) {

unsigned int(8) uri\_initialization\_data[];

}

class URIMetaSampleEntry() extends MetaDataSampleEntry (’urim‘) {

URIbox the\_label;

URIInitBox init; // optional

MPEG4BitRateBox (); // optional

}

// Visual Sequences

class PixelAspectRatioBox extends Box(‘pasp’){

unsigned int(32) hSpacing;

unsigned int(32) vSpacing;

}

class CleanApertureBox extends Box(‘clap’){

unsigned int(32) cleanApertureWidthN;

unsigned int(32) cleanApertureWidthD;

unsigned int(32) cleanApertureHeightN;

unsigned int(32) cleanApertureHeightD;

unsigned int(32) horizOffN;

unsigned int(32) horizOffD;

unsigned int(32) vertOffN;

unsigned int(32) vertOffD;

}

class ColourInformationBox extends Box(‘colr’){

unsigned int(32) colour\_type;

if (colour\_type == ‘nclx’) /\* on-screen colours \*/

{

unsigned int(16) colour\_primaries;

unsigned int(16) transfer\_characteristics;

unsigned int(16) matrix\_coefficients;

unsigned int(1) full\_range\_flag;

unsigned int(7) reserved = 0;

}

else if (colour\_type == ‘rICC’)

{

ICC\_profile; // restricted ICC profile

}

else if (colour\_type == ‘prof’)

{

ICC\_profile; // unrestricted ICC profile

}

}

class VisualSampleEntry(codingname) extends SampleEntry (codingname){

unsigned int(16) pre\_defined = 0;

const unsigned int(16) reserved = 0;

unsigned int(32)[3] pre\_defined = 0;

unsigned int(16) width;

unsigned int(16) height;

template unsigned int(32) horizresolution = 0x00480000; // 72 dpi

template unsigned int(32) vertresolution = 0x00480000; // 72 dpi

const unsigned int(32) reserved = 0;

template unsigned int(16) frame\_count = 1;

string[32] compressorname;

template unsigned int(16) depth = 0x0018;

int(16) pre\_defined = -1;

// other boxes from derived specifications

CleanApertureBox clap; // optional

PixelAspectRatioBox pasp; // optional

}

// Audio Sequences

class AudioSampleEntry(codingname) extends SampleEntry (codingname){

const unsigned int(32)[2] reserved = 0;

template unsigned int(16) channelcount = 2;

template unsigned int(16) samplesize = 16;

unsigned int(16) pre\_defined = 0;

const unsigned int(16) reserved = 0 ;

template unsigned int(32) samplerate = { default samplerate of media}<<16;

}

aligned(8) class SampleDescriptionBox (unsigned int(32) handler\_type)

extends FullBox('stsd', 0, 0){

int i ;

unsigned int(32) entry\_count;

for (i = 1 ; i <= entry\_count ; i++){

switch (handler\_type){

case ‘soun’: // for audio tracks

AudioSampleEntry();

break;

case ‘vide’: // for video tracks

VisualSampleEntry();

break;

case ‘hint’: // Hint track

HintSampleEntry();

break;

case ‘meta’: // Metadata track

MetadataSampleEntry();

break;

}

}

}

#### 8.5.2.3 Semantics

version是一个指定此方框的版本的整数

entry\_count是一个整数，它给出了下表中的条目数

SampleEntry是适当的样品输入。

data\_reference\_index是一个整数，它包含用于检索与使用此示例描述的示例关联的数据的数据引用的索引。数据引用存储在Data Reference Boxes中。索引的范围从1到数据引用的数量。

ChannelCount 是频道的数量，如1（单声道）或2（立体声）

SampleSize以位为单位，取默认值为16

SampleRate是用16.16定点数(hi.lo)表示的采样率

resolution字段表示图像的分辨率为每英寸像素，作为一个固定的16.16个数字

frame\_count表示在每个示例中存储了压缩视频的多少帧。默认值为1，每个样本一帧；每个样本的多个帧可能超过1

Compressorname是一个名称，用于提供信息的目的。它的格式化为一个固定的32字节字段，第一个字节设置为要显示的字节数，然后是可显示数据的字节数，然后填充以完成总共32个字节（包括大小字节）。该字段可以设置为0。

depth取以下值之一

0x0018-图像是颜色的，没有alpha

width和height是由这个示例描述所描述的流的最大视觉宽度和高度，以像素为单位

hSpacing, vSpacing：定义一个像素的相对宽度和高度；

cleanApertureWidthN, cleanApertureWidthD:定义视频图像的精确清晰光圈宽度的分数数

cleanApertureHeightN, cleanApertureHeightD:定义视频图像的精确干净孔径高度的分数数

horizOffN, horizOffD: 一个分数，定义清洁孔径中心减去（宽度-1）的水平偏移量/2。通常为0。

vertOffN, vertOffD: 一个分数，它定义了清洁孔径中心减去（高度-1）的垂直偏移量/2。通常为0

content\_encoding -是一个以UTF-8字符的空结尾字符串，并提供一个MIME类型来标识定时元数据的内容编码。它的定义与本规范中的信息输入相同。如果不存在（提供了一个空字符串），则不会对定时元数据进行编码。此字段的一个示例是“application/zip”。请注意，目前不存在针对BiM[ISO/IEC23001-1]和TeM[ISO/IEC15938-1]的MIME类型。因此，应使用实验性MIME类型的“application/x-BiM”和“text/x-TeM”来识别这些编码机制。

namespace-为定时XML元数据提供模式的命名空间。这对于识别元数据的类型，如gBSD或AQoS[MPEG-21-7]和使用XML感知的编码机制，如BiM进行解码所必需的。

schema\_location——可选地提供一个URL来查找与名称空间相对应的模式。这是通过XML感知的编码机制，如BiM来解码定时元数据所必需的。

mime\_format-提供了一个MIME类型，用来标识定时元数据的内容格式。此字段的示例是“text/html”和“text/plain”

bufferSizeDB以字节为单位给出了基本流的解码缓冲区的大小

MaxBitrate在1秒的任何窗口中给出以位/秒为单位的最大速率。

avgBitrate给出了整个表示过程中位/秒的平均速率。

theURI是根据6.2.4中的规则格式化的URI；

uri\_initialization\_data是一种不透明的数据，其形式在URI表单的文档中定义。

colour\_type：所提供的颜色信息类型的指示。对于colour\_type‘nclx’：这些字段正是在ISO/IEC29199-2的A.7.2中为PTM\_COLOR\_INFO()定义的四个字节，但请注意，全范围标志在这里处于不同的位位置

ICC\_profile：提供ISO 15076-1或ICC. 1:10 0中定义的ICC配置文件。

### 8.5.3 Degradation Priority Box

#### 8.5.3.1 Definition

Box Type: ‘stdp’

Container: Sample Table Box (‘stbl’).

Mandatory: No.

Quantity: Zero or one.

这个box包含了每个样品的降解优先级。这些值存储在表中，每个示例对应一个值。表的大小，sample\_count取自Sample Size Box(“stsz”)中的sample\_count。由此导出的规范定义了优先级字段的确切含义和可接受范围。

#### 8.5.3.2 Syntax

aligned(8) class DegradationPriorityBox

extends FullBox(‘stdp’, version = 0, 0) {

int i;

for (i=0; i < sample\_count; i++) {

unsigned int(16) priority;

}

}

#### 8.5.3.3 Semantics

version 是指定此box的版本的整数。

priority是为每个样本的退化优先级的整数。

### 8.5.4 Sample Scale Box

(empty sub-clause)

## 8.6 Track Time Structures

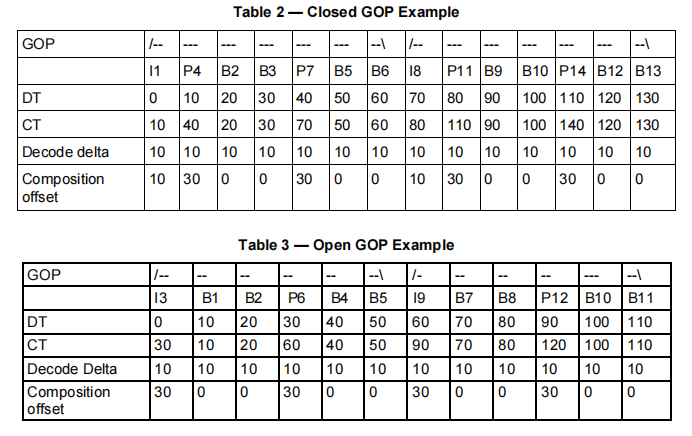
### 8.6.1 Time to Sample Boxes

#### 8.6.1.1 Definition

样本的组成时间(CT)和解码时间(DT)来自于Time to Sample Boxes，其中有两种类型。解码时间在Decoding Time to Sample Box中定义，给出连续解码时间之间的时间增量。合成时间在Composition Time to Sample Box中作为解码时间的合成时间偏移。如果track中每个样品的合成时间和解码时间相同，则只需要解码到sample box的时间；sample box的合成时间必须不存在。

time to sample boxes必须给所有样本非零持续时间，最后一个可能除外。“stts”box中的持续时间是严格的正的（非零），除了最后一个条目，它可能是零。这个规则来源于一个流中没有两个时间戳可能是相同的规则。当向流中添加样本时，必须非常小心，之前的样本可能需要建立一个非零持续时间，以遵守此规则。如果最后一个示例的持续时间不确定，请使用一个任意的小值和一个“驻留”编辑。

在下面的例子中，有一个I、P和B帧的序列，每个帧的解码时间增量为10。样本存储如下，其解码时间增量和组合时间偏移量的指示值(实际的CT和DT可供参考)。重新排序的发生是因为预测的P帧必须在双向预测的B帧之前被解码。一个样本的DT值总是前面两个样本的增量之和。请注意，解码增量的总数是媒体在此track中的持续时间。



#### 8.6.1.2 Decoding Time to Sample Box

8.6.1.2.1 Definition

Box Type: ‘stts’

Container: Sample Table Box (‘stbl’)

Mandatory: Yes

Quantity: Exactly one

这个box包含一个紧凑版本的表，允许从解码时间到样本号进行索引。其他表给出了从样本编号中得到的样本大小和指针。表中的每个条目都给出了具有相同时间增量的连续样本的数量，以及这些样本的增量。通过添加deltas，可以建立一个完整的样本时间映射。

Decoding Time to Sample Box包含解码时间delta：DT(n+1)=DT(n)+STTS(n)，其中STTS(n)是样本n的（未压缩）表条目。

样本条目通过解码时间戳进行排序；因此，delta都是非负的。

DT轴有一个零原点；DT(i)=SUM(用于delta(j)的j=0到i-1)，所有delta的总和给出了track中介质的长度（没有映射到整个时间尺度，也不考虑任何编辑列表）。

如果Edit List Box的CT值非空（非零），则提供初始CT值。

##### 8.6.1.2.2 Syntax

aligned(8) class TimeToSampleBox

extends FullBox(’stts’, version = 0, 0) {

unsigned int(32) entry\_count;

int i;

for (i=0; i < entry\_count; i++) {

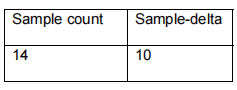
unsigned int(32) sample\_count;

unsigned int(32) sample\_delta;

}

}

例如，对于表2，该条目将为：



##### 8.6.1.2.3 Semantics

version 是指定此框的版本的整数

entry\_count-是一个整数，给出下表中条目的数量。

sample\_count-是一个整数，它计算具有给定持续时间的连续样本的数量。

sample\_delta-是一个整数，它在媒体的时间尺度上给出了这些样本的增量。

#### 8.6.1.3 Composition Time to Sample Box

##### 8.6.1.3.1 Definition

Box Type: ‘ctts’

Container: Sample Table Box (‘stbl’)

Mandatory: No

Quantity: Zero or one

此box提供了解码时间和合成时间之间的偏移量。在这个box的版本0中，解码时间必须小于合成时间，偏移量表示为无符号数字，这样CT(n)=DT(n)+CTTS(n)，其中CTTS(n)是样本n的（未压缩）表条目。在这个box的版本1中，组合时间轴和解码时间轴仍然是相互派生的，但是偏移量是有签名的。建议对于计算的合成时间戳，恰好有一个值为0（零）的戳。

对于任何一个版本的box，每个样本都必须有一个唯一的组合时间戳值，即两个样本的时间戳永远不能相同。

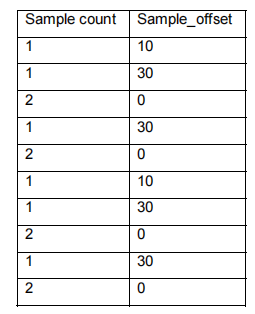
时间0可能没有帧，未指定处理（系统可能显示第一帧更长，或合适的填充颜色）。

当使用此box的版本1时，CompositionToDecodeBox也可以出现在示例表中，以关联组合和解码时间线。当需要向后兼容或与一组未知读取器兼容时，应尽可能使用此box的版本0。在此box的任何一个版本中，特别是在版本0下，如果希望媒体在跟踪时间0开始，并且第一个媒体样本的合成时间不是0，则可以使用编辑列表将媒体“移”到时间0。

样品表的组成时间是可选的，只有在任何样品的DT和CT不同时才必须存在。

Hint track不使用此box。

例如，在表2中



##### 8.6.1.3.2 Syntax

aligned(8) class CompositionOffsetBox

extends FullBox(‘ctts’, version = 0, 0) {

unsigned int(32) entry\_count;

int i;

if (version==0) {

for (i=0; i < entry\_count; i++) {

unsigned int(32) sample\_count;

unsigned int(32) sample\_offset;

}

}

else if (version == 1) {

for (i=0; i < entry\_count; i++) {

unsigned int(32) sample\_count;

signed int(32) sample\_offset;

}

}

}

8.6.1.3.3 Semantics

version 是指定此框的版本的整数。

entry\_count是一个整数，它给出了下表中的条目数。

sample\_count是一个整数，它计算具有给定偏移量的连续样本的数量。

sample\_offset是一个整数，它给出了CT和DT之间的偏移量，这样，CT(n)=DT(n)+CTTS(n)。

#### 8.6.1.4 Composition to Decode Box

##### 8.6.1.4.1 Definition

Box Type: ‘cslg’

Container: Sample Table Box (‘stbl’)

Mandatory: No

Quantity: Zero or one

当使用符号组合偏移量时，此box可用于关联组合和解码时间线，并处理符号组合偏移量引入的一些歧义。

请注意，所有这些字段都适用于整个媒体（不仅仅是任何编辑选择的媒体）。建议任何明确或暗示的编辑，不要选择没有映射到样本的合成时间线的任何部分。例如，如果最小的合成时间为1000，则从0到媒体持续时间的默认编辑将保留从0到1000的周期。在这些情况下，播放者的行为以及在这个间隔内的组成是不确定的。建议计算出的最小CTS为零，或匹配第一次编辑的开始时间。

track中最后一个样本的合成持续时间可能是（通常是）不明确或不清楚的；合成结束时间的字段可以用来澄清这种模糊性，并随着合成开始时间，为track建立一个清晰的合成持续时间。

当Composition to Decode Box包含在Sample Table Box中时，它只记录了movie box中样本的合成和解码时间关系，而不包括任何后续的movie片段。

##### 8.6.1.4.2 Syntax

class CompositionToDecodeBox extends FullBox(‘cslg’, version=0, 0) {

signed int(32) compositionToDTSShift;

signed int(32) leastDecodeToDisplayDelta;

signed int(32) greatestDecodeToDisplayDelta;

signed int(32) compositionStartTime;

signed int(32) compositionEndTime;

}

##### 8.6.1.4.3 Semantics

compositionToDTSShift：如果将此值添加到合成时间（由DTS的CTS偏移量计算），则所有样本的CTS保证大于或等于DTS，尊重指定轮廓/级别所隐含的缓冲模型；如果显示增量为正或零，则该字段可以为0；否则至少为（-十二点到显示增量）

leastDecodeToDisplayDelta：此track中的CompositionTimeToSample box 中的最小组合偏移量

greatestDecodeToDisplayDelta：此track中CompositionTimeToSample box中最大的组合偏移量

compositionStartTime：此track介质中任何样品的最小计算合成时间(CTS)

compositionEndTime：在该track介质中计算合成时间(CTS)最大的样品的合成时间加上合成持续时间；如果该字段取值为0，则合成结束时间未知。

### 8.6.2 Sync Sample Box

#### 8.6.2.1 Definition

Box Type: ‘stss’

Container: Sample Table Box (‘stbl’)

Mandatory: No

Quantity: Zero or one

这个box提供了流中同步示例的紧凑标记。该表格按样本数严格递增的顺序排列。

如果不存在sync sample box，则每个示例都是一个sync sample。

#### 8.6.2.2 Syntax

aligned(8) class SyncSampleBox

extends FullBox(‘stss’, version = 0, 0) {

unsigned int(32) entry\_count;

int i;

for (i=0; i < entry\_count; i++) {

unsigned int(32) sample\_number;

}

}

#### 8.6.2.3 Semantics

version 是指定此框的版本的整数。

entry\_count是一个整数，它给出了下表中的条目数。如果entry\_count为零，则流中没有同步示例，下表为空。

sample\_number给出了在流中是同步样本的样本的数量。

### 8.6.3 Shadow Sync Sample Box

#### 8.6.3.1 Definition

Box Type: ‘stsh’

Container: Sample Table Box (‘stbl’)

Mandatory: No

Quantity: Zero or one

shadow sync table提供了一组可选的同步示例，可以在查找或用于类似目的时使用。在正常的向前播放中，他们都被忽略了。

ShadowSyncTable中的每个条目都由一对样本编号组成。第一个条目（(shadowed-samplenumber）表示将为其定义shadow sync的样本数量。这应该总是一个非同步的样本（例如，帧差异）。第二个样本号（sync-sample-number）表示在shadowed-sample-number或之前需要同步样本时可以使用的同步样本号（即关键帧）。

ShadowSyncBox中的条目应根据shadowed-sample-number字段进行排序。

shadow sync samples通常被放置在正常播放期间没有出现的track区域（通过编辑列表编辑出来），尽管这不是必需的。shadow sync table可以被忽略，如果它被忽略（尽管可能不是最佳状态），track将正确地播放（和搜索）。

ShadowSyncSample替换而不是增强其shadow的样本（即下一个发送的样本是shadowed-sample-number+1）。shadow sync sample被视为它发生在shadow样本的时候，并具有shadow样本的持续时间。

如果一个shadow sample也被用作正常回放的一部分，或者多次被用作一个shaow，那么隐藏和传输可能会变得更加复杂。在这种情况下，hint track可能需要单独的shadow syncs，所有这些shadow sync都可以从media track中的一个shadow sync中获得它们的媒体数据，以允许不同的时间戳等。需要放在他们的header中。

#### 8.6.3.2 Syntax

aligned(8) class ShadowSyncSampleBox

extends FullBox(‘stsh’, version = 0, 0) {

unsigned int(32) entry\_count;

int i;

for (i=0; i < entry\_count; i++) {

unsigned int(32) shadowed\_sample\_number;

unsigned int(32) sync\_sample\_number;

}

}

#### 8.6.3.3 Semantics

version 是指定此框的版本的整数。

entry\_count-是一个整数，给出下表中条目的数量。

shadowed\_sample\_number-给出有替代同步示例的示例的数量。

sync\_sample\_number-给出了替代同步样本的数量。

### 8.6.4 Independent and Disposable Samples Box

#### 8.6.4.1 Definition

Box Types: ‘sdtp’

Container: Sample Table Box (‘stbl’)

Mandatory: No

Quantity: Zero or one

此可选表回答了关于示例依赖性的三个问题：

1. 这些样本是否依赖于其他样本（例如它是I图片）？
2. 难道没有其他的样本依赖于这个样本吗？
3. 此示例是否在此时时刻包含多个（冗余）数据编码（可能具有不同的依赖关系）？

如果没有此表：

1. sync sample table（部分）回答了第一个问题；在大多数视频编解码器中，I-sicks也是同步点，
2. 其他样本对这个样本的依赖性是未知的。
3. 冗余编码的存在是未知的。

当执行“trick”模式时，比如快进，可以使用第一条信息来定位独立的可解码的样本。类似地，在执行随机访问时，可能需要定位以前的同步样本或随机访问恢复点，并从随机访问恢复点的同步样本或预滚起始点前滚到所需的点。在向前滚动时，不需要检索或解码其他不依赖的样本。

“sample\_is\_depended\_on”的值与冗余编码的存在无关。但是，冗余编码可能与主编码有不同的依赖关系；如果冗余编码可用，“sample\_depends\_on”的值只记录主编码。

一个leading 样本（通常是视频中的图片）是相对于一个参考样本定义的，这是标记为“sample\_depends\_on”的样本(一个I图片)。一个leading 样本在参考样本之前有一个合成时间，也可能有一个对参考样本之前的样本的解码依赖。因此，例如，如果回放和解码从参考样本开始，那些标记为领先的样本将不需要，也可能无法解码。因此，一个leading 样本本身不能被标记为没有依赖性

Tracks handler\_type不是“vide”，“soun”，“hint”或“auxv”，如果另一个样本sample\_depends\_on=2或另一个样本标记为“Sync Sample”已经处理，除非另有指定，一个样本标记sample\_depends\_on=2，和sample\_has\_redundancy=1可以丢弃，其持续时间添加到前面的持续时间，维护后续样本的时间。

表的大小sample\_count取自样本大小框(“stsz”)或压缩样本大小框(“stz2”)中的sample\_count。

#### 8.6.4.2 Syntax

aligned(8) class SampleDependencyTypeBox

extends FullBox(‘sdtp’, version = 0, 0) {

for (i=0; i < sample\_count; i++){

unsigned int(2) is\_leading;

unsigned int(2) sample\_depends\_on;

unsigned int(2) sample\_is\_depended\_on;

unsigned int(2) sample\_has\_redundancy;

}

}

is\_leading采用以下四个值之一：

0: 这个样本的主要性质是未知的；

1：这个样本是一个leading 样本，在引用的i-图片之前有依赖性（因此不可解码）；

2：这个样本不是一个leading 样本；

3：这个样本是一个leading 样本，在引用的i图片之前没有依赖性（因此是可解码的）；

sample\_depends\_on采用以下四个值之一：

0: 此样本的依赖性未知；

1：该样本确实依赖于其他样本（不是I图片）；

2：此样本不依赖于其他样本（I图片）；

3：保留

sample\_is\_depended\_on采用以下四个值之一：

0: 其他样品对该样品的依赖性未知；

1：其他样品可能依赖于这个（非一次性）；

2：没有其他样品依赖于这个（一次性）；

3：保留

sample\_has\_redundancy取以下四个值之一：

0：本例中是否存在冗余编码

1：本例中存在冗余编码

2：本例中没有冗余编码

3：保留

### 8.6.5 Edit Box

#### 8.6.5.1 Definition

Box Type: ‘edts’

Container: Track Box (‘trak’)

Mandatory: No

Quantity: Zero or one

Edit Box将表示时间线映射到存储在文件中的媒体时间线。该Edit Box是一个用于显示编辑列表的容器。

该Edit Box是可选的。在没有这个box的情况下，有一个关于这些时间线的隐式一对一映射，并且track的表示从表示的开始就开始了。一个空的编辑被用于偏移一个track的开始时间。

#### 8.6.5.2 Syntax

aligned(8) class EditBox extends Box(‘edts’) {

}

### 8.6.6 Edit List Box

#### 8.6.6.1 Definition

Box Type: ‘elst’

Container: Edit Box (‘edts’)

Mandatory: No

Quantity: Zero or one

此box包含一个显式的时间轴贴图。每个条目定义了跟踪时间线的一部分：通过映射媒体时间线的一部分，或者通过指示“empty”时间，或者通过定义一个“dwell”，其中媒体中的单个时间点被保存一段时间。

注：编辑不限于采样时间。这意味着当输入编辑时，可能需要(a)备份到一个同步点，并从那里预滚动，然后(b)注意第一个样本的持续时间——如果编辑在正常持续时间内输入，它可能会被截断。如果这是音频，该帧可能需要被解码，然后完成最后的切片。同样，编辑中最后一个示例的持续时间可能需要切片。

tracks（流）的起始偏移量由初始空编辑表示。例如，从一开始就播放一段曲目30秒，但在presentation开始10秒时，我们有以下编辑列表：

Entry-count = 2

Segment-duration = 10 seconds

Media-Time = -1

Media-Rate = 1

Segment-duration = 30 seconds (可能是整个track的长度)

Media-Time = 0 seconds

Media-Rate = 1

一个非空的编辑可以插入在初始movie中没有出现的媒体时间轴的一部分，并且只出现在后续的movie片段中。特别是在碎片movie文件的空初始电影中（当没有媒体样本时），此编辑的segment\_duration可能为零，因此编辑为movie和随后的movie片段提供从媒体合成时间到movie呈现时间的偏移。当使用组合偏移量时，建议使用这种编辑来为第一个呈现的样本建立一个表示时间为0。

例如，如果第一个合成帧的合成时间为20，那么将媒体时间从20以后映射到电影时间0以后的编辑将显示：

Entry-count = 1

Segment-duration = 0

Media-Time = 20

Media-Rate = 1

#### 8.6.6.2 Syntax

aligned(8) class EditListBox extends FullBox(‘elst’, version, 0) {

unsigned int(32) entry\_count;

for (i=1; i <= entry\_count; i++) {

if (version==1) {

unsigned int(64) segment\_duration;

int(64) media\_time;

} else { // version==0

unsigned int(32) segment\_duration;

int(32) media\_time;

}

int(16) media\_rate\_integer;

int(16) media\_rate\_fraction = 0;

}

}

#### 8.6.6.3 Semantics

version是一个整数，它指定此框的版本（0或1）

entry\_count是一个整数，它给出了下表中的条目数

segment\_duration是一个整数，它以Movie Header Box中的时间尺度为单位来指定这个编辑段的持续时间

media\_time是一个整数，包含此编辑段媒体内的开始时间（以媒体时间尺度单位，以合成时间单位）。如果此字段设置为-1，则它为空的编辑。track中的最后一次编辑永远不能是空的编辑。Move Header Box中的持续时间与track的持续时间之间的任何差异都表示为结尾的隐式空编辑。

media\_rate指定播放与此编辑段对应的媒体的相对速率。如果此值为0，则编辑指定“dwell”：在分段期间显示媒体时间的媒体。否则，该字段应包含值1。

## 8.7 Track Data Layout Structures

### 8.7.1 Data Information Box

#### 8.7.1.1 Definition

Box Type: ‘dinf’

Container: Media Information Box (‘minf’) or Meta Box (‘meta’)

Mandatory: Yes (required within ‘minf’ box) and No (optional within ‘meta’ box)

Quantity: Exactly one

数据信息框包含声明track中媒体信息的位置的对象。

#### 8.7.1.2 Syntax

aligned(8) class DataInformationBox extends Box(‘dinf’) {

}

8.7.2 Data Reference Box

8.7.2.1 Definition

Box Types: ‘url ‘, ‘urn ‘, ‘dref’

Container: Data Information Box (‘dinf’)

Mandatory: Yes

Quantity: Exactly one

数据引用对象包含一个数据引用表（通常是URL），用来声明在presentation中使用的媒体数据的位置。示例描述中的数据引用索引将此表中的条目与track中的示例联系起来。一个track可以以这种方式分成几个源。

如果设置的标志指示数据与此box在同一文件中，则输入字段中不提供任何字符串（甚至是空字符串）。

DataReferenceBox内的DataEntryBox应为DataEntryUrnBox或DataEntryUrlBox。

注意，虽然计数是32位，但项目的数量通常要少得多，并且受到样本表中的引用索引只有16位的限制

当具有数据条目的标记集指示媒体数据位于同一文件中的文件被分割为段进行传输时，此标志的值不会改变，因为该文件在传输操作后（逻辑上）重新组装。

#### 8.7.2.2 Syntax

aligned(8) class DataEntryUrlBox (bit(24) flags)

extends FullBox(‘url ’, version = 0, flags) {

string location;

}

aligned(8) class DataEntryUrnBox (bit(24) flags)

extends FullBox(‘urn ’, version = 0, flags) {

string name;

string location;

}

aligned(8) class DataReferenceBox

extends FullBox(‘dref’, version = 0, 0) {

unsigned int(32) entry\_count;

for (i=1; i <= entry\_count; i++) {

DataEntryBox(entry\_version, entry\_flags) data\_entry;

}

}

#### 8.7.2.3 Semantics

version 是一个指定此box的版本的整数

entry\_count是一个计算实际条目的整数

entry\_version是一个整数，它指定了输入格式的版本

entry\_flags是一个带有标志的24位整数；其中定义了一个标志(x000001)，这意味着媒体数据与包含此数据引用的Movie Box在同一文件中。

data\_entry是一个URL或URN条目。名称是URN，是URN条目中必需的。位置是一个URL，在URL条目中是必需的，在URN条目中是可选的，其中它提供了一个查找具有给定名称的资源的位置。每个都是一个使用UTF-8编码的以null结尾的字符串。如果设置了自包含的标志，则使用URL表单，且不存在字符串；该box以入口标志字段终止。URL类型应该是提供文件的服务(例如，文件类型的URL、http、ftp等)，以及哪些服务最好也允许随机访问。允许使用相对url，相对于包含Movie Box的文件 ，这个Movie Box包含此数据引用。

### 8.7.3 Sample Size Boxes

#### 8.7.3.1 Definition

Box Type: ‘stsz’, ‘stz2’

Container: Sample Table Box (‘stbl’)

Mandatory: Yes

Quantity: Exactly one variant must be present

这个box包含示例计数和一个表，其中给出了每个示例的字节大小。这允许媒体数据本身被unframed。介质中的样品总数总是在样品计数中表示。

样本大小box有两种变体。第一个变体有一个固定大小的32位字段，用于表示样本大小；它允许为一个track中的所有样本定义一个恒定的大小。第二种变体允许较小大小的字段，以便在大小变化但较小时节省空间。必须有其中一个box；为了达到最大程度的兼容性，首选使用第一个版本。

注：样本大小一般不禁止，但必须使用样本所属的样本条目定义的编码系统。

#### 8.7.3.2 Sample Size Box

##### 8.7.3.2.1 Syntax

aligned(8) class SampleSizeBox extends FullBox(‘stsz’, version = 0, 0) {

unsigned int(32) sample\_size;

unsigned int(32) sample\_count;

if (sample\_size==0) {

for (i=1; i <= sample\_count; i++) {

unsigned int(32) entry\_size;

}

}

}

##### 8.7.3.2.2 Semantics

version是一个指定此方框的版本的整数

sample\_size是指定默认示例大小的整数。如果所有示例的大小都相同，则此字段将包含该大小值。如果此字段设置为0，则样本具有不同的大小，这些大小将存储在样本大小表中。如果此字段不是0，则它指定常量的样本大小，并且不包含任何数组。

sample\_count是一个整数，它给出了track中的样本数；如果样本大小为0，则它也是下表中的条目数。

entry\_size是一个整数，指定一个样本的大小，并按其编号进行索引。

#### 8.7.3.3 Compact Sample Size Box

##### 8.7.3.3.1 Syntax

aligned(8) class CompactSampleSizeBox extends FullBox(‘stz2’, version = 0, 0) {

unsigned int(24) reserved = 0;

unisgned int(8) field\_size;

unsigned int(32) sample\_count;

for (i=1; i <= sample\_count; i++) {

unsigned int(field\_size) entry\_size;

}

}

##### 8.7.3.3.2 Semantics

version是一个指定此方框的版本的整数

field\_size是一个整数，指定下表中项的大小，取值4、8或16。如果使用值4，则每个字节包含两个值：条目[i]<<4+条目[i+1]；如果大小不填充整数字节，最后一个字节填充零。

sample\_count是一个整数，它给出了下表中的条目数

entry\_size是一个整数，指定一个样本的大小，并按其编号进行索引。

### 8.7.4 Sample To Chunk Box

#### 8.7.4.1 Definition

Box Type: ‘stsc’

Container: Sample Table Box (‘stbl’)

Mandatory: Yes

Quantity: Exactly one

媒体数据中的样本被分组为数据块。块可以有不同的大小，块中的样本可以有不同的大小。此表可用于查找包含一个示例及其位置和关联的示例描述的块。

该表的编码方式很紧凑。每个条目给出了具有相同特征的块的第一个块的索引。通过从前一个条目中减去一个条目，您可以计算出这次运行中有多少块。您可以通过乘以每个块的适当样本来将其转换为样本计数。

#### 8.7.4.2 Syntax

aligned(8) class SampleToChunkBox

extends FullBox(‘stsc’, version = 0, 0) {

unsigned int(32) entry\_count;

for (i=1; i <= entry\_count; i++) {

unsigned int(32) first\_chunk;

unsigned int(32) samples\_per\_chunk;

unsigned int(32) sample\_description\_index;

}

}

#### 8.7.4.3 Semantics

version 是一个指定此方框的版本的整数

entry\_count是一个整数，它给出了下表中的条目数

first\_chunk是一个整数，它给出了这个运行中第一个块的索引，这些块共享相同的样本和样本描述索引；track中第一个块的索引的值为1(此box的第一个记录中的first\_chunk字段的值为1，标识第一个示例映射到第一个块)。

samples\_per\_chunk是一个整数，它给出了每个这些数据块中的样本数

sample\_description\_index是一个整数，它给出了描述该块中的示例的示例条目的索引。索引的范围从1到“示例描述”box中的示例条目数量

8.7.5 Chunk Offset Box

#### 8.7.5.1 Definition

Box Type: ‘stco’, ‘co64’

Container: Sample Table Box (‘stbl’)

Mandatory: Yes

Quantity: Exactly one variant must be present

chunk offset table给出了所包含的文件中的每个块的索引。有两种变体，允许使用32位或64位偏移量。后者在管理非常大的presentation时非常有用。这些变体最多有一个将出现在示例表的任何单个实例中。

Offsets是文件偏移，而不是文件中任何框的偏移（例如媒体数据框）。这允许引用没有任何box结构的文件中的媒体数据。这也意味着在前面构建具有元数据（电影盒）的自包含ISO文件时必须小心，因为movie box的大小会影响媒体数据的块偏移。

**8.7.5.2 Syntax**

aligned(8) class ChunkOffsetBox

extends FullBox(‘stco’, version = 0, 0) {

unsigned int(32) entry\_count;

for (i=1; i <= entry\_count; i++) {

unsigned int(32) chunk\_offset;

}

}

aligned(8) class ChunkLargeOffsetBox

extends FullBox(‘co64’, version = 0, 0) {

unsigned int(32) entry\_count;

for (i=1; i <= entry\_count; i++) {

unsigned int(64) chunk\_offset;

}

}

8.7.5.3 Semantics

version是一个指定此方框的版本的整数

entry\_count是一个整数，它给出了下表中的条目数

chunk\_offset是一个32或64位整数，它将块的开始偏移到包含媒体文件的偏移量。

### 8.7.6 Padding Bits Box

#### 8.7.6.1 Definition

Box Type: ‘padb’

Container: Sample Table (‘stbl’)

Mandatory: No

Quantity: Zero or one

在某些流中，媒体样本并不占据由样本大小给出的所有字节位，并在末尾填充到一个字节边界。在某些情况下，需要在外部记录所使用的填充位数。此表提供了该信息。

#### 8.7.6.2 Syntax

aligned(8) class PaddingBitsBox extends FullBox(‘padb’, version = 0, 0) {

unsigned int(32) sample\_count;

int i;

for (i=0; i < ((sample\_count + 1)/2); i++) {

bit(1) reserved = 0;

bit(3) pad1;

bit(1) reserved = 0;

bit(3) pad2;

}

}

#### 8.7.6.3 Semantics

sample\_count-计算track中的样本数；它应该与其他表中的计数相匹配

Pad1-从0到7的值，表示样本（i\*2）+1结尾的位数

Pad2-从0到7的值，表示样本（i\*2）+2结尾的位数

### 8.7.7 Sub-Sample Information Box

8.7.7.1 Definition

Box Type: ‘subs’

Container: Sample Table Box (‘stbl’) or Track Fragment Box (‘traf’)

Mandatory: No

Quantity: Zero or one

此box名为“*Sub-Sample Information box*”box，设计用于包含子样本信息。

子样本是一个样本的连续字节范围。子样本的具体定义应提供给给定的编码系统（例如，ISO/IEC14496-10，高级视频编码）。如果没有这样的具体定义，此box不得应用于使用该编码系统的样品。

如果subsample\_count对于任何条目为0，那么这些样本没有子样本信息，也没有数组。该表采用稀疏编码；该表通过记录每个条目之间的样本数的差异来确定哪些样本具有子样本结构。表中的第一个条目记录了具有子样本信息的第一个样本的样本数。

注意：可以将subsample\_priority和可丢弃值组合起来，这样当subsample\_priority小于某个值时，可丢弃值就被设置为1。然而，由于不同的系统可能使用不同规模的优先级值，因此要分离它们是安全的，就可以为可丢弃的子样本提供一个干净的解决方案。

#### 8.7.7.2 Syntax

aligned(8) class SubSampleInformationBox

extends FullBox(‘subs’, version, 0) {

unsigned int(32) entry\_count;

int i,j;

for (i=0; i < entry\_count; i++) {

unsigned int(32) sample\_delta;

unsigned int(16) subsample\_count;

if (subsample\_count > 0) {

for (j=0; j < subsample\_count; j++) {

if(version == 1)

{

unsigned int(32) subsample\_size;

}

else

{

unsigned int(16) subsample\_size;

}

unsigned int(8) subsample\_priority;

unsigned int(8) discardable;

unsigned int(32) reserved = 0;

}

}

}

}

#### 8.7.7.3 Semantics

version是一个整数，它指定此框的版本（本规范中的0或1）

entry\_count是一个整数，它给出了下表中的条目数。

sample\_delta是一个整数，它指定具有子样本结构的样本的样本数。它被编码为所需的样本数与前一个条目中指示的样本数之间的差值。如果当前条目是第一个条目，则该值表示具有子样本信息的第一个样本的样本号，即该值为样本数与零（0）之间的差值。

subsample\_count是一个指定当前样本的子样本数量的整数。如果没有子样本结构，则该字段的值为0。

subsample\_size是一个整数，它指定当前子示例的大小，单位是以字节为单位。

subsample\_priority是一个整数，指定每个子样本的退化优先级。subsample\_priority值越高，表示子样本对解码质量很重要，并且对解码质量的影响越大。

discardable等于0表示子样本需要解码当前样本，而等于1表示子样本不需要解码当前样本，但可以用于增强，例如，子样本由补充增强信息(SEI)消息组成。

### 8.7.8 Sample Auxiliary Information Sizes Box

#### 8.7.8.1 Definition

Box Type: ‘saiz’

Container: Sample Table Box (‘stbl’) or Track Fragment Box ('traf')

Mandatory: No

Quantity: Zero or More

每个样本样本辅助信息可以与样本数据本身存储在同一文件的任何地方；对于自包含的媒体文件，这通常在MediaData box或来自派生规范的box中。它存储在多个块中，每个块的样本数量，以及块的数量，匹配主样本数据的块数量，或者在movie sample table（或a movie fragment）中所有样本的单个块中存储(b)。包含在单个块（或跟踪运行）中的所有示例的示例辅助信息将连续存储（类似于示例数据）。

样本辅助信息存在时，总是与与其关联的样本存储在相同一个文件中，因为它们共享相同的数据引用(“dref”)结构。但是，这些数据可能位于此文件中的任何地方，使用辅助信息偏移量(“saio”)来指示数据的位置。

是否允许或需要样品辅助信息可以由使用的品牌或编码格式指定。样本辅助信息的格式由aux\_info\_type确定。如果省略了aux\_info\_type和aux\_info\_type\_parameter，那么aux\_info\_type的隐含值在转换内容的情况下是(a)，如保护内容，保护方案信息框中包含的scheme\_type，或者(b)示例输入类型。aux\_info\_type\_parameter的默认值为0。aux\_info\_type的某些值可能被限制为仅用于特定的track类型。一个track可以具有多个不同类型的样本辅助信息流。这些类型已在登记机关登记。

虽然aux\_info\_type决定了辅助信息的格式，但当其aux\_info\_type\_parameter的值不同时，可以使用多个具有相同格式的辅助信息流。必须指定特定aux\_info\_type值的aux\_info\_type\_parameter的语义，同时指定特定aux\_info\_type值的语义和隐含的辅助信息格式。

这个box提供了每个样本的辅助信息的大小。对于此box的每个实例，必须有一个匹配的aux\_info\_type和aux\_info\_type\_parameter值，样本辅助信息偏移框，为此辅助信息提供偏移信息。

注：关于使用样本辅助信息与其他机制的讨论，见附录C.8。

#### 8.7.8.2 Syntax

aligned(8) class SampleAuxiliaryInformationSizesBox

extends FullBox(‘saiz’, version = 0, flags)

{

if (flags & 1) {

unsigned int(32) aux\_info\_type;

unsigned int(32) aux\_info\_type\_parameter;

}

unsigned int(8) default\_sample\_info\_size;

unsigned int(32) sample\_count;

if (default\_sample\_info\_size == 0) {

unsigned int(8) sample\_info\_size[ sample\_count ];

}

}

#### 8.7.8.3 Semantics

aux\_info\_type是一个标识示例辅助信息类型的整数。包含box中最多应出现一个与aux\_info\_type和aux\_info\_type\_parameter值相同的box。

aux\_info\_type\_parameter标识具有相同的aux\_info\_type值并与相同的track相关联的辅助信息的“流”。aux\_info\_type\_parameter的语义由aux\_info\_type的值决定。

default\_sample\_info\_size是一个整数，指定所有指示的样本具有相同的样本辅助信息大小的样本辅助信息大小。如果大小发生变化，则该字段应为零。

sample\_count是一个整数，它给出了定义了大小的样本数。对于出现在样本表框中的样本辅助信息大小框，这必须与样本大小框或紧凑样本大小框中的sample\_count相同或小于。对于出现在track片段框中的Sample Auxiliary Information Sizes，这必须与Track Fragment box的Track Fragment Run boxes中的sample\_count条目的总和相同或小于。如果这小于样本数，则为初始样本提供辅助信息，而其余的样本没有相关的辅助信息。

sample\_info\_size以字节为单位表示示例辅助信息的大小。这可能是零，以表示没有相关辅助信息的样本。

### 8.7.9 Sample Auxiliary Information Offsets Box

#### 8.7.9.1 Definition

Box Type: ‘saio’

Container: Sample Table Box (‘stbl’) or Track Fragment Box ('traf')

Mandatory: No

Quantity: Zero or More

有关示例辅助信息的介绍，请参阅Sample Auxiliary Information Size Box的定义。

这个box提供了样本辅助信息的位置信息，其方式类似于样本数据的块偏移量。

#### 8.7.9.2 Syntax

aligned(8) class SampleAuxiliaryInformationOffsetsBox

extends FullBox(‘saio’, version, flags)

{

if (flags & 1) {

unsigned int(32) aux\_info\_type;

unsigned int(32) aux\_info\_type\_parameter;

}

unsigned int(32) entry\_count;

if ( version == 0 ) {

unsigned int(32) offset[ entry\_count ];

}

else {

unsigned int(64) offset[ entry\_count ];

}

}

**8.7.9.3 Semantics**

aux\_info\_type 和 aux\_info\_type\_parameter 在 SampleAuxiliaryInformationSizesBox 被定义。

entry\_count给出了下表中的条目数。对于出现在示Sample Table Box中的Sample Auxiliary Information Offsets box，它必须等于一个或在Chunk Offset Box或Chunk Large Offset Box中的entry\_count字段的值。对于出现在Track Fragment box中的Sample Auxiliary Information Offsets Box，这必须等于一个或等于Track Fragment Box中的Track Fragment Run boxs的数量。

offset给出了每个块或Track Fragment Run的样本辅助信息的文件中的位置。如果entry\_count为1，则所有块或运行的示例辅助信息在文件的块或运行顺序中是连续的。当在Sample Table Box中时，偏移量是绝对的。在track fragment box中，此值相对于同一track fragment中的track fragment header box（“tfhd”）建立的基本偏移量（参见8.8.14）。

## 8.8 Movie Fragments

8.8.1 Movie Extends Box

8.8.1.1 Definition

Box Type: ‘mvex’

Container: Movie Box (‘moov’)

Mandatory: No

Quantity: Zero or one

此box警告读取者，此文件中可能存在Movie Fragment Boxes。为了知道tracks中的所有样本，必须按顺序找到这些Movie Fragment Boxes并进行扫描，并将它们的信息逻辑地添加到在Movie box中找到的信息中。

在附录a中有一个对Movie fragments的叙述性介绍。

#### 8.8.1.2 Syntax

aligned(8) class MovieExtendsBox extends Box(‘mvex’){

}

### 8.8.2 Movie Extends Header Box

#### 8.8.2.1 Definition

Box Type: ‘mehd’

Container: Movie Extends Box(‘mvex’)

Mandatory: No

Quantity: Zero or one

Movie Extends Header是可选的，它提供了片段movie的整体持续时间，包括片段。如果不存在此box，则必须通过检查每个片段来计算总的持续时间。

8.8.2.2 Syntax

aligned(8) class MovieExtendsHeaderBox extends FullBox(‘mehd’, version, 0) {

if (version==1) {

unsigned int(64) fragment\_duration;

} else { // version==0

unsigned int(32) fragment\_duration;

}

}

#### 8.8.2.3 Semantics

fragment\_duration是一个整数，它声明了整个movie的呈现长度，包括片段（在Movie Header Box中指示的时间尺度中）。该字段的值对应于最长音轨的持续时间，包括movie片段。如果MP4文件是实时创建的，比如在直播中使用，则不太可能提前知道fragment\_duration，这个框可能被省略。

### 8.8.3 Track Extends Box

#### 8.8.3.1 Definition

Box Type: ‘trex’

Container: Movie Extends Box (‘mvex’)

Mandatory: Yes

Quantity: Exactly one for each track in the Movie Box

这将设置movie片段所使用的默认值。通过以这种方式设置默认值，可以保存在每个Track Fragment Box中的空间和复杂性。

示例片段中的示例标记字段(这里的default\_sample\_flags和Track Fragment Header Box中的default\_sample\_flags，以及Track Fragment Run Box中的sample\_flags和first\_sample\_flags)被编码为32位值。其结构如下：

bit(4) reserved=0;

unsigned int(2) is\_leading;

unsigned int(2) sample\_depends\_on;

unsigned int(2) sample\_is\_depended\_on;

unsigned int(2) sample\_has\_redundancy;

bit(3) sample\_padding\_value;

bit(1) sample\_is\_non\_sync\_sample;

unsigned int(16) sample\_degradation\_priority;

is\_leading, sample\_depends\_on, sample\_is\_depended\_on 和 sample\_has\_redundancy 值被定义为在独立和Disposable Samples Box中的记录。

标志sample\_is\_non\_sync\_sample提供了与同步示例表相同的信息[8.6.2]。当此值设置为0时，它与样本不在movie片段中，并在同步样本表中用一个条目标记（或者，如果所有样本都是同步样本，则没有同步样本表）相同。

sample\_padding\_value被定义为填充位表。sample\_degradation\_priority被定义为降级优先级表。

#### 8.8.3.2 Syntax

aligned(8) class TrackExtendsBox extends FullBox(‘trex’, 0, 0){

unsigned int(32) track\_ID;

unsigned int(32) default\_sample\_description\_index;

unsigned int(32) default\_sample\_duration;

unsigned int(32) default\_sample\_size;

unsigned int(32) default\_sample\_flags

}

#### 8.8.3.3 Semantics

track\_id标识track，这应该是Movie Box中的track的track ID

default\_这些字段设置了在track片段中使用的默认值。

### 8.8.4 Movie Fragment Box

#### 8.8.4.1 Definition

Box Type: ‘moof’

Container: File

Mandatory: No

Quantity: Zero or more

Movie片段及时延长了放映时间。他们提供了以前应该放在Movie Box里的信息。像往常一样，如果实际样本在同一个文件中，那么实际样本就在Media Data Boxes中。数据引用索引在示例描述中，因此可以构建增量presentations，其中媒体数据位于包含 Movie Box的文件之外的文件中。

Movie片段框是一个顶级box（即，对应于 Movie Box 和Media Data boxes）。它包含一个Movie Fragment Header Box，然后是一个或多个Track Fragment Boxes。

注意：没有要求任何特定的movie片段扩展movie header中出现的所有tracks，对movie片段引用的媒体数据的位置没有限制。然而，派生的规范可能会做出这样的限制。

#### 8.8.4.2 Syntax

aligned(8) class MovieFragmentBox extends Box(‘moof’){

}

### 8.8.5 Movie Fragment Header Box

8.8.5.1 Definition

Box Type: ‘mfhd’

Container: Movie Fragment Box ('moof')

Mandatory: Yes

Quantity: Exactly one

movie fragment header包含一个序列号，作为一个安全检查。序列号通常从1开始，并且必须为文件中的每个movie片段按照它们出现的顺序而增加。这允许读取者验证序列的完整性；构建一个片段失序的文件是错误的。

注意：没有要求序列号是连续的，只有给定movie片段中的值大于前面任何movie片段中的值。

#### 8.8.5.2 Syntax

aligned(8) class MovieFragmentHeaderBox

extends FullBox(‘mfhd’, 0, 0){

unsigned int(32) sequence\_number;

}

#### 8.8.5.3 Semantics

sequence\_number是这个片段的序数，依次递增。

### 8.8.6 Track Fragment Box

#### 8.8.6.1 Definition

Box Type: ‘traf’

Container: Movie Fragment Box ('moof')

Mandatory: No

Quantity: Zero or more

在movie片段中有一组track片段，每个track有零或更多。track片段依次包含0个或多个track运行，每个track运行记录了该track的连续样本运行。在这些结构中，许多字段都是可选的，并且可以默认使用。

可以使用这些结构向track添加“空时间”，并添加样本。例如，空插入可以用于做沉默抑制的audio track。

#### 8.8.6.2 Syntax

aligned(8) class TrackFragmentBox extends Box(‘traf’){

}

### 8.8.7 Track Fragment Header Box

#### 8.8.7.1 Definition

Box Type: ‘tfhd’

Container: Track Fragment Box ('traf')

Mandatory: Yes

Quantity: Exactly one

每个movie片段可以向每个track添加零个或多个片段；一个track片段可以添加零个或多个连续的样本运行。跟踪片段头设置了用于这些样本运行的信息和默认值。

在tf\_flags中定义了以下标志：

0x000001存在基本数据偏移量：表示存在基本数据偏移量字段。这为每个track运行中的数据偏移量提供了一个显式的锚点（请参见下文）。如果不提供，电影片段中第一个track的基本数据偏移量是包围的Movie Fragment Box的第一个字节的位置，对于第二个和后续的跟踪片段，默认值是前一个片段定义的数据的末尾。以这种方式“继承”偏移的片段必须使用相同的数据引用（即，这些track的数据必须在同一个文件中）。

0x000002示例描述索引存在：表示存在此字段，该字段在此片段中覆盖了在跟踪扩展框中设置的默认值。

0x000008默认样本持续时间存在

0x000010默认样本大小存在

0x000020默认样本标志存在

0x010000持续时间为空：这表示默认样本持续时间或跟踪扩展box中默认持续时间中提供的持续时间为空，即此时间间隔没有样本。在movie box中包含编辑列表和空持续时间片段的presentation是错误的。

0x020000默认基本模式：如果存在基本数据偏移量为零，则表示此跟踪片段的基本数据偏移量是包围的Movie Fragment Box的第一个字节的位置。“iso5” brand 需要支持默认的标志，不得在iso5之前的 brand 或兼容 brand 中使用。

注意：使用默认基准点标志破坏了文件格式早期brand的兼容性，因为它对偏移计算的锚点设置与早期不同。因此，当“文件类型”框中包含早期的brand时，不能设置默认的基本模式标志。

#### 8.8.7.2 Syntax

aligned(8) class TrackFragmentHeaderBox

extends FullBox(‘tfhd’, 0, tf\_flags){

unsigned int(32) track\_ID;

// all the following are optional fields

unsigned int(64) base\_data\_offset;

unsigned int(32) sample\_description\_index;

unsigned int(32) default\_sample\_duration;

unsigned int(32) default\_sample\_size;

unsigned int(32) default\_sample\_flags

}

#### 8.8.7.3 Semantics

base\_data\_offset在计算数据偏移量时要使用的基本偏移量

### 8.8.8 Track Fragment Run Box

#### 8.8.8.1 Definition

Box Type: ‘trun’

Container: Track Fragment Box ('traf')

Mandatory: No

Quantity: Zero or more

在Track Fragment Box中，有零或多个Track Run Boxes。如果在tf\_flags中设置了持续时间为空的标志，则没有track运行。一个track运行为一个track记录了一组连续的样本。

可选字段的数量由在标志的下字节中设置的位数以及在标志的第二个字节中设置的位数的记录大小决定。应遵循此程序，以便于定义新的字段。

如果不存在数据偏移，则此运行的数据在上一次运行的数据之后立即开始，或者如果这是track片段中的第一次运行，则在跟踪片段头定义的base-data-offset处，如果存在数据偏移，则相对于跟踪片段头中建立的基本数据偏移。

已定义了以下标志：

0x000001数据偏移-存在。

0x000004存在第一个示例标志；此操作仅覆盖第一个示例的默认标志。这使得可以记录一组帧，其中第一个帧是键，其余帧是不同帧，而无需为每个示例提供显式标记。如果使用此标志和字段，则不得出现样本标志。

0x000100 sample-duration-present：表示每个样本都有自己的持续时间，否则使用默认值。

0x000200 sample-size-present：每个样本都有自己的大小，否则使用默认值。

0x000400 sample-flags-present：每个示例都有自己的标志，否则使用默认值。

0x000800 sample-composition-time-offsets-presen：每个样本都有一个合成时间偏移量（例如，用于MPEG中的I/P/B视频）。

composition time-to-sample box和track run box中的composition offset 可以是有符号的或无符号的。在composition time-to-sample box中给出的关于使用有符号的组合偏移量的建议也适用于这里。

#### 8.8.8.2 Syntax

aligned(8) class TrackRunBox

extends FullBox(‘trun’, version, tr\_flags) {

unsigned int(32) sample\_count;

// the following are optional fields

signed int(32) data\_offset;

unsigned int(32) first\_sample\_flags;

// all fields in the following array are optional

{

unsigned int(32) sample\_duration;

unsigned int(32) sample\_size;

unsigned int(32) sample\_flags

if (version == 0)

{ unsigned int(32) sample\_composition\_time\_offset; }

else

{ signed int(32) sample\_composition\_time\_offset; }

}[ sample\_count ]

}

#### 8.8.8.3 Semantics

sample\_count在此运行中添加的示例数；以及下表中的行数（行可以为空）

data\_offset被添加到在跟踪片段头中建立的隐式或显式data\_offset中。

first\_sample\_flags仅为此运行的第一个示例提供了一组标志。

### 8.8.9 Movie Fragment Random Access Box

#### 8.8.9.1 Definition

Box Type: ‘mfra’

Container: File

Mandatory: No

Quantity: Zero or one

Movie Fragment Random Access Box (“mfra”)提供了一个表，它可以帮助读取者使用movie片段在文件中查找同步样本。它包含为提供信息的每个track的track fragment random access box（可能不是所有tracks）。它通常被放置在文件的末尾或文件的附近；Movie Fragment Random Access Box 中的最后一个框提供了从Movie Fragment Random Access Box 中获得的长度字段的副本。读取者可能会试图找到这个盒子通过检查最后32位的文件，或向后扫描文件的Movie Fragment Random Access Offset Box和使用大小信息，看看是否定位Movie Fragment Random Access Box的开始。

这个box只提供了一个关于同步样本在哪里的hint；movie片段本身是明确的。建议读取者在定位和使用此box作为创建文件后的修改时要注意，这可能会导致指针或同步示例声明不正确。

#### 8.8.9.2 Syntax

aligned(8) class MovieFragmentRandomAccessBox

extends Box(‘mfra’)

{

}

8.8.10 Track Fragment Random Access Box

#### 8.8.10.1 Definition

Box Type: ‘tfra’

Container: Movie Fragment Random Access Box (‘mfra’)

Mandatory: No

Quantity: Zero or one per track

每个条目都包含同步示例的位置和表示时间。请注意，表中并不需要列出track中的每个同步示例。

没有这个box并不意味着所有的样本都是同步的样本。“trun”、“traf”和“trex”中的随机访问信息。

#### 8.8.10.2 Syntax

aligned(8) class TrackFragmentRandomAccessBox

extends FullBox(‘tfra’, version, 0) {

unsigned int(32) track\_ID;

const unsigned int(26) reserved = 0;

unsigned int(2) length\_size\_of\_traf\_num;

unsigned int(2) length\_size\_of\_trun\_num;

unsigned int(2) length\_size\_of\_sample\_num;

unsigned int(32) number\_of\_entry;

for(i=1; i <= number\_of\_entry; i++){

if(version==1){

unsigned int(64) time;

unsigned int(64) moof\_offset;

}else{

unsigned int(32) time;

unsigned int(32) moof\_offset;

}

unsigned int((length\_size\_of\_traf\_num+1) \* 8) traf\_number;

unsigned int((length\_size\_of\_trun\_num+1) \* 8) trun\_number;

unsigned int((length\_size\_of\_sample\_num+1) \* 8) sample\_number;

}

}

#### 8.8.10.3 Semantics

track\_ID是一个标识track\_ID的整数。

length\_size\_of\_traf\_num表示traf\_number字段的字节长度减去1。

length\_size\_of\_trun\_num表示trun\_number字段的字节长度减去1。

length\_size\_of\_sample\_num表示sample\_number字段的字节长度减去1。

number\_of\_entry是一个整数，它给出了这个track的条目的数量。如果此值为零，则表示每个示例都是一个同步示例，并且后面没有表条目。

time 是32或64位整数，表示同步样本以相关track的mdhd为单位定义的表示时间。

moof\_offset是32或64位整数，给出了此条目中使用的“moof”的偏移量。偏移量是文件开始和“moof”开始之间的字节偏移量。

traf\_number表示包含同步样本的“traf”号。每个“moof”中的数字范围为1(第一个“traf”编号为1)。

trun\_number表示包含同步示例的“trun”编号。每个“traf”中的数字范围为1。

sample\_number表示同步样本的样本号。每个“trun”中的数字范围为1。

### 8.8.11 Movie Fragment Random Access Offset Box

#### 8.8.11.1 Definition

Box Type: ‘mfro’

Container: Movie Fragment Random Access Box (‘mfra’)

Mandatory: Yes

Quantity: Exactly one

Movie Fragment Random Access Offset Box提供了enclosing Movie Fragment Random Access Box的长度字段的副本。它被放置在该box的最后，因此size字段也在enclosing Movie Fragment Random Access Box之后。当Movie Fragment Random Access Box也最后出现在文件中时，这允许它的位置更方便。这里的size字段必须是正确的。然而，Movie Fragment Random Access Box是否存在，以及它的位置是否在文件中的最后，都不能得到保证。

#### 8.8.11.2 Syntax

aligned(8) class MovieFragmentRandomAccessOffsetBox

extends FullBox(‘mfro’, version, 0) {

unsigned int(32) size;

}

#### 8.8.11.3 Semantics

size是一个整数，给出了包围的“mfra” box的字节数。此字段放置在包围box的最后一个，以帮助读取者从文件的末尾进行扫描，以找到“mfra”框。

### 8.8.12 Track fragment decode time

#### 8.8.12.1 Definition

Box Type: `tfdt’

Container: Track Fragment box (‘traf’)

Mandatory: No

Quantity: Zero or one

Track Fragment Base Media Decode Time Box提供了在track片段中按解码顺序排列的第一个样本在媒体时间线上测量的绝对解码时间。例如，在文件中执行随机访问时，这很有用；不需要将以前片段中所有前面样本的采样时间求和以找到此值（其中样本持续时间是解码时间到样本box中的增量和前面跟踪运行中的sample\_durations）。

如果存在Track Fragment Base Media Decode Time Box，应放置在Track Fragment Header Box 之后和第一个Track Fragment Run box之前。

注意：解码时间轴是一个媒体时间轴，是在媒体时间到presentation时间的任何显式或隐含的映射之前建立的，例如通过一个编辑列表或类似的结构。

#### 8.8.12.2 Syntax

aligned(8) class TrackFragmentBaseMediaDecodeTimeBox

extends FullBox(‘tfdt’, version, 0) {

if (version==1) {

unsigned int(64) baseMediaDecodeTime;

} else { // version==0

unsigned int(32) baseMediaDecodeTime;

}

}

#### 8.8.12.3 Semantics

version是一个整数，它指定此box的版本（此规范中的0或1）。

baseMediaDecodeTime是一个整数，等于媒体中所有早期样本的解码持续时间之和，以媒体的时间尺度表示。它不包括在包围的（enclosing）track片段中添加的样本。

8.8.13 Level Assignment Box

#### 8.8.13.1 Definition

Box Type: `leva’

Container: Movie Extends Box (`mvex’)

Mandatory: No

Quantity: Zero or one

级别指定文件的子集。映射到n级的样本可能依赖于任何m级的样本，其中m<=n，不依赖于任何p级的样本，其中p>n。例如，级别可以根据时间级别来指定(例如SVC或MVC的temporal\_id)。

无法为初始movie指定级别。当Level Assignment box存在时，它将应用于初始movie之后的所有movie片段

对于Level Assignment box的上下文，一个分数被定义为包含一个或多个Movie Fragment boxes 和相关的Media Data boxes，可能只包括最后一个Media Data box的初始部分。在一个部分内，每个级别的数据应该连续出现。分数内的数据应以水平值的递增顺序显示。分数中的所有数据都应按等级分配。

注：在DASH（ISO/IEC23009-1）中，Subsegment Index box 中索引的每个子段都是一部分。

Level Assignment box 提供从特性(如可伸缩性层)到级别的映射。可以通过track、track中的子track或track的样本分组来指定特性。

当padding\_flag等于1 这表明一个符合分数可以通过在分数中连接任何正整数级别数，并填充最后一个媒体数据框的零字节，直到最后一个媒体数据框头部中表示的完整大小来形成。例如，当以下条件为真时，可以将padding\_flag设置为等于1：

* 每个部分包含两个或多个同一视频位流的AVC、SVC或MVC[ISO/IEC14496-15]track。
* 一个部分的每个track的样本都是连续的，并在一个Media Data box中按解码顺序排列
* 第一个AVC、SVC或MVC水平的样本包含提取器NAL单元，以包括来自同一部分的其他水平的视频编码NAL单元。

#### 8.8.13.2 Syntax

aligned(8) class LevelAssignmentBox extends FullBox(‘leva’, 0, 0)

{

unsigned int(8) level\_count;

for (j=1; j <= level\_count; j++) {

unsigned int(32) track\_id;

unsigned int(1) padding\_flag;

unsigned int(7) assignment\_type;

if (assignment\_type == 0) {

unsigned int(32) grouping\_type;

}

else if (assignment\_type == 1) {

unsigned int(32) grouping\_type;

unsigned int(32) grouping\_type\_parameter;

}

else if (assignment\_type == 2) {} // no further syntax elements needed

else if (assignment\_type == 3) {} // no further syntax elements needed

else if (assignment\_type == 4) {

unsigned int(32) sub\_track\_id;

}

// other assignment\_type values are reserved

}

}

#### 8.8.13.3 Semantics

level\_count指定每个分数被分组到的级别数。level\_count应大于或等于2。

track\_id对于循环项j，指定了赋给第j级的track的track标识符。  。

padding\_flag等于1表示，可以通过连接一个分数内的任何正整数级别，并在最后一个Media Data box的头中指示的完整大小来形成符合分数。padding\_flag等于0的语义是不能保证的。

assignment\_type表示用于指定对一个级别的分配的机制。保留大于4的assignment\_type值，而其他值的语义指定如下。assignment\_types的序列被限制为类型2或3的零或以上的集合，后面是一个类型的零或多个。

* 0:用样本组来指定等级，即某一样本组的不同样本组描述指标对应的样本在确定的track内处于不同等级; 其他track则不受影响，必须将所有数据精确地保存在一个level中;
* 1:跟assignment\_type 0一样，除了是由参数化的样本组赋值;
* 2、3：级别通过track分配（参见子分段索引框，请参阅这些级别处理的差异）
* 4: 各自的级别包含一个sub-track的样本。sub-track通过sub-track box指定，其他track不受影响，其所有数据必须只在一个级别内；

grouping\_type 与grouping\_type\_parameter如果存在则指定用于将Sample Group Description box中的样本组描述条目映射到级别的示例分组。级别n包含映射到样本组描述项的样本，在Sample Group Description box中，如果存在，则与本box中提供的grouping\_type和grouping\_type\_parameter值相同

sub\_track\_id指定在循环项j中由sub\_track\_id标识的sub-track被映射到级别j。

### 8.8.14 Sample Auxiliary Information in Movie Fragments

当Movie Fragment box中出现样本辅助信息（8.7.8和8.7.9）时，Sample Auxiliary Information Offsets Box中的偏移量与Track Fragment Run box中的data\_offset相同，也就是说，它们相对于为该trackc片段建立的任何基本数据偏移量。如果使用了movie片段相对寻址（在track片段头中没有提供基本数据偏移量），并且存在辅助信息，则还必须在该跟踪片段头的标志中设置default\_base\_is\_moof标志。

如果只提供一个偏移量，则片段中所有track运行的样本辅助信息将连续存储，否则必须为每个track运行提供一个偏移量。

如果字段default\_sample\_info\_size在其中的一个box中不为零，那么对于已识别的样本，辅助信息的大小是恒定的。

此外，如果：

* 这个box出现在movie box里，
* 而且在movie box里的box里的default\_sample\_info\_size是非零的，
* 而在movie片段中sample auxiliary information sizes box，

那么辅助信息对于movie片段中的每个样本也有相同的常数大小；这样就没有必要在movie片段中重复这个box了。

## 8.9 Sample Group Structures

### 8.9.1 Introduction

此子句指定了一种用于表示track中的样本的分区的通用机制。样本分组是根据分组标准，将一个track中的每个样本分配为一个样本组的成员。样本分组中的样本组不限于连续的样本，也可能包含非相邻的样本。由于一个track中的样本可能有多个样本分组，因此每个样本分组都有一个类型字段来表示分组的类型。例如，一个文件可能包含同一track的两个样本组：一个基于对层的样本分配，另一个基于对子序列的样本分配。

样本组由两个链接的数据结构表示：（1）一个SampleToGroup box表示样本分配给样本组；（2）一个SampleGroupDescription box包含一个样本组项，描述组的属性。根据不同的分组标准，SampleToGroup and SampleGroupDescription boxes可能有多个实例。它们通过一个用于指示分组类型的类型字段来区分。

特定分组类型的分组可以使用示例中的参数来进行组映射；如果是这样，则必须使用该组来记录该参数的含义。其中一个例子可以记录多个视频流中的同步点；组定义可能是“是一个I帧”，组参数可能是每个流的标识符。由于每个流的示例到组框发生一次，因此它现在既紧凑，又分别通知读取者每个流。

使用这些表的一个示例是表示对图层的样本分配。在这种情况下，每个样本组代表一个层，其中一个SampleToGroup box的实例描述了一个样本属于哪个层。

### 8.9.2 Sample to Group Box

#### 8.9.2.1 Definition

Box Type: ‘sbgp’

Container: Sample Table Box (‘stbl’) or Track Fragment Box (‘traf’)

Mandatory: No

Quantity: Zero or more.

此表可用于查找样本所属的组以及该样本组的关联描述。该表被紧凑地编码，每个条目都给出了具有相同样本组描述符的样本运行中的第一个样本的索引。样本组描述ID是一个索引，它引用了一个SampleGroupDescription box，其中包含描述每个样本组的特征的条目。

如果一个track中的样本有多个样本分组，则可能存在此box的多个实例。样本组框中的每个实例都有一个类型代码，用于区分不同的样本分组。在一个track内，最多应该有一个具有特定分组类型的这个box的实例。相关的样本组说明应表示分组类型的相同值。

只有在需要分组类型参数时才应该使用此框的版本1。

#### 8.9.2.2 Syntax

aligned(8) class SampleToGroupBox

extends FullBox(‘sbgp’, version, 0)

{

unsigned int(32) grouping\_type;

if (version == 1) {

unsigned int(32) grouping\_type\_parameter;

}

unsigned int(32) entry\_count;

for (i=1; i <= entry\_count; i++)

{

unsigned int(32) sample\_count;

unsigned int(32) group\_description\_index;

}

}

#### 8.9.2.3 Semantics

version是一个整数，它指定此框的版本，即0或1。

grouping\_type是一个整数，它标识样本分组的类型（即用于形成样本组的标准），并将其链接到与分组类型相同的值的样本组描述表。对于一个track，最多出现一次grouping\_type(如果使用grouping\_type\_parameter)。

grouping\_type\_parameter是分组的子类型的指示

entry\_count是一个整数，它给出了下表中的条目数。

sample\_count是一个整数，它表示具有相同样本组描述符的连续样本数。如果这个box中的样本计数之和小于总样本计数，那么读取者应该有效地扩展它，将剩余的样本与no group相关联。如果这个box中的总数大于其他地方记录的sample\_count，这是一个错误，那么读取者的行为将不被定义。

group\_description\_index是一个整数，它给出了描述该组中的样本的样本组条目的索引。索引的范围从1到样本组描述框中的样本组条目数，或者取值0表示该样本不属于此类型的组的成员。

### 8.9.3 Sample Group Description Box

#### 8.9.3.1 Definition

Box Type: ‘sgpd’

Container: Sample Table Box (‘stbl’) or Track Fragment Box (‘traf’)

Mandatory: No

Quantity: Zero or more, with one for each Sample to Group Box.

这个描述表提供了关于样本组的特征的信息。描述性信息是定义或描述样本组所需的任何其他信息。

如果一个track中的样本有多个样本分组，则可能存在此box的多个实例。SampleGroupDescription box的每个实例都有一个类型代码，用于区分不同的样本分组。在一个track内，最多应该有一个具有特定分组类型的这个box的实例。关联的SampleToGroup应为分组类型指定相同的值。

输入计数后，信息存储在样本组描述框中。定义了一个抽象的条目类型，样本组应定义派生的类型，以表示每个样本组的描述。对于视频track，一个抽象的图像类型和类似的类型一起用于音频和hint track。

注意：在条目版本为0的版本中，示例组描述条目的的基类既不是box，也没有大小。因此，不赞成使用版本0的条目。在定义派生类时，请确保它们具有固定的大小，或者用长度字段明确表示该大小。不建议使用隐含的大小（例如，通过解析数据来实现），因为这会使扫描数组变得困难。

#### 8.9.3.2 Syntax

// Sequence Entry

abstract class SampleGroupDescriptionEntry (unsigned int(32) grouping\_type)

{

}

abstract class VisualSampleGroupEntry (unsigned int(32) grouping\_type) extends

SampleGroupDescriptionEntry (grouping\_type)

{

}

abstract class AudioSampleGroupEntry (unsigned int(32) grouping\_type) extends

SampleGroupDescriptionEntry (grouping\_type)

{

}

abstract class HintSampleGroupEntry (unsigned int(32) grouping\_type) extends

SampleGroupDescriptionEntry (grouping\_type)

{

}

aligned(8) class SampleGroupDescriptionBox (unsigned int(32) handler\_type)

extends FullBox('sgpd', version, 0){

unsigned int(32) grouping\_type;

if (version==1) { unsigned int(32) default\_length; }

unsigned int(32) entry\_count;

int i;

for (i = 1 ; i <= entry\_count ; i++){

if (version==1) {

if (default\_length==0) {

unsigned int(32) description\_length;

}

}

switch (handler\_type){

case ‘vide’: // for video tracks

VisualSampleGroupEntry (grouping\_type);

break;

case ‘soun’: // for audio tracks

AudioSampleGroupEntry(grouping\_type);

break;

case ‘hint’: // for hint tracks

HintSampleGroupEntry(grouping\_type);

break;

}

}

}

#### 8.9.3.3 Semantics

version是一个指定此方框的版本的整数。

grouping\_type是一个整数，用于标识与此示例组描述相关联的SampleToGroup box。

entry\_count是一个整数，它给出了下表中的条目数。

default\_length表示每个组条目的长度(如果长度是常量)，如果长度是变量，则为0 。

description\_length表示单个组条目的长度，如果条目不同，因此default\_length为0

### 8.9.4 Representation of group structures in Movie Fragments

对movie片段中的样本组结构的支持是通过使用SampleToGroup box提供的，该box的容器是Track Fragment Box(“traf”）。此框的定义、语法和语义在第8.9.2款中有所规定。

SampleToGroup Box 可用于查找跟踪片段中的样本所属的组以及该样本组的关联描述。该表被紧凑地编码，每个条目都给出了具有相同样本组描述符的样本运行中的第一个样本的索引。样本组描述ID是一个索引，是SampleGroupDescription Box的引用，它包含描述每个样本组特征的条目，并将在SampleTableBox中显示。

如果一个track片段中的样本有更多的一个样本分组，则可能存在SampleToGroupBox的多个实例。SampleToGroup Box的每个实例都有一个类型代码，用于区分不同的样本分组。相关的样本组说明应表示分组类型的相同值。

track片段中任何SampleToGroup Box 中表示的样本总数必须与所有track片段运行中的样本总数相匹配。每个SampleToGroup Box 记录了相同样本的不同分组

零个或更多个SampleGroupDescription box也可能出现在一个trackc片段框中。这些定义是movie box中track的示例表中提供的定义的附加定义。Movie片段中的组定义也可以从同一movie片段中被引用和使用。

在该movie片段的样本toGroup box中，在同一片段中定义的组的组描述索引从0x10001开始，即索引值1，值1位于前16位。这意味着在movie box中的示例表中，这个跟踪和分组类型必须有少于65536个组定义。

当更改movie片段的大小或删除它们时，需要将这些片段-本地组定义合并到movie box中的定义中，或合并到新的movie片段中，并对SampleToGroup box(es) 中的索引编号进行相应的调整。建议在此过程中，不要在任何SampleGroupDescription box中进行相同的（因此是重复的）定义，而是合并重复的定义并相应地调整索引。

8.10 User Data

8.10.1 User Data Box

#### 8.10.1.1 Definition

Box Type: ‘udta’

Container: Movie Box (‘moov’) or Track Box (‘trak’)

Mandatory: No

Quantity: Zero or one

此box包含声明有关包含box及其数据（表示或跟踪）的用户信息的对象。

User Data Box是一个用来提供信息的用户数据的容器箱。这个用户数据被格式化为一组具有更特定box类型的box，它们更精确地声明它们的内容。

本规范中仅定义了一个版权声明。

#### 8.10.1.2 Syntax

aligned(8) class UserDataBox extends Box(‘udta’) {

}

### 8.10.2 Copyright Box

#### 8.10.2.1 Definition

Box Type: ‘cprt’

Container: User data box (‘udta’)

Mandatory: No

Quantity: Zero or more

Copyright box包含一个版权声明，当包含在movie box中时，或当包含在一个track中时，适用于整个track。可能会有多个使用不同语言代码的版权框。

8.10.2.2 Syntax

aligned(8) class CopyrightBox

extends FullBox(‘cprt’, version = 0, 0) {

const bit(1) pad = 0;

unsigned int(5)[3] language; // ISO-639-2/T language code

string notice;

}

#### 8.10.2.3 Semantics

language声明了以下文本的语言代码。有关三个字符代码集，请参见ISO639-2/T。每个字符都被打包为其ASCII值与0x60之间的差值。代码仅限于三个小写字母，因此这些值严格地是正的。

notice是一个UTF-8或UTF-16编码的以null结尾的字符串，给出版权通知。如果使用UTF-16，该字符串应以BYTE ORDER MARK(0xFEFF)开始，以区别于UTF-8字符串。这个标记并不构成最终字符串的一部分。

### 8.10.3 Track Selection Box

#### 8.10.3.1 Introduction

存储在文件中的典型presentation文稿，每个媒体类型包含一个替代组：一个用于视频，一个用于音频，等等。这样的文件可能包括几个视频track，尽管在任何时间点，只应该播放或streamed其中一个。这是通过将所有的视频track分配给同一替代组来实现的。（替代组的定义见第8.3.2款。）

另一组中的所有曲目都是媒体选择的候选曲目，但在一次会议期间在其中一些曲目之间切换可能没有意义。例如，可以允许在不同比特率的视频track之间切换，并保持帧大小，但不允许在不同帧大小的track之间切换。以同样的方式，可能希望在不同视频编解码器或不同音频语言的track之间进行选择，但不能切换。

除了选择组之外，通过分配切换组来解决选择和切换trackc之间的区别。一个替代组可能包含一个或多个开关组。替代组中的所有trackc都是媒体选择的候选对象，而交换组中的track也可用于在会话期间进行切换。不同的开关组代表不同的操作点，如不同的帧大小、高/低质量等。

对于不可扩展的位流的情况，在一个交换机组中可能包含几个track。这也适用于非分层可伸缩的位流，比如传统的AVC流。

通过用属性标记track，就可以描述它们的特征。每个track都可以标记一个属性列表，这些属性可以用来描述特定交换组中的track或区分属于不同交换组的track。

#### 8.10.3.2 Definition

Box Type: ‘tsel’

Container: User Data Box (‘udta’)

Mandatory: No

Quantity: Zero or One

track选择框包含在其修改的track的用户数据框中。

#### 8.10.3.3 Syntax

aligned(8) class TrackSelectionBox

extends FullBox(‘tsel’, version = 0, 0) {

template int(32) switch\_group = 0;

unsigned int(32) attribute\_list[]; // to end of the box

}

#### 8.10.3.4 Semantics

switch\_group是一个指定一组或多个track集合的整数。如果此字段为0（默认值）或没有曲目选择框，则没有关于该曲目是否可以在播放或流媒体期间用于切换的信息。如果这个整数不是0，则可用于彼此之间切换的track应相同。属于同一交换机组的track应属于同一交换机组。一个交换机组可能只有一个成员。

attribute\_list是一个属性的列表。此列表中的属性应用作同一备用或交换组中track的描述或区分标准。每个区分属性都与一个指向区分track的字段或信息的指针相关联。

#### 8.10.3.5 Attributes

描述了以下属性：

|  |  |  |
| --- | --- | --- |
| *Name* | *Attribute* | *Description* |
| Temporal scalability | ‘tesc’ | 该track可以按时间顺序进行缩放。 |
| Fine-grain SNR scalability | ‘fgsc’ | 该track可以根据质量来进行调整。 |
| Coarse-grain SNR scalability | ‘cgsc’ | 该track可以根据质量来进行调整。 |
| Spatial scalability | ‘spsc’ | 该track可以进行空间缩放。 |
| Region-of-interest scalability | ‘resc’ | 该track可以按感兴趣的区域范围进行扩展 |
| View scalability | ‘vwsc’ | 该track可以根据视图的数量进行缩放。 |

区别对待以下属性：

|  |  |  |
| --- | --- | --- |
| *Name* | *Attribute* | *Description* |
| Codec | ‘cdec’ | 样本输入（在媒体track的样本说明框中） |
| Screen size | ‘scsz’ | 视觉样本条目的宽度和高度字段。 |
| Max packet size | ‘mpsz’ | RTP提示示例条目中的Maxpacketsize字段 |
| Media type | ‘mtyp’ | Handler box中的Handlertype（为媒体track） |
| Media language | ‘mela’ | Language field in Media Header box |
| Bitrate | ‘bitr’ | trackc中样本的总大小除以track header box中的持续时间 |
| Frame rate | ‘frar’ | track中的样本数除以track header box中的持续时间 |
| Number of views | ‘nvws’ | sub track中的视图数 |

描述性属性描述了它们所修改的track，而区别对待的属性则区分了属于同一交替组或切换组的trackc。区别对待的属性的指针指示将track与具有相同属性的其他track区分开来的信息的位置。

## 8.11 Metadata Support

一个公共的基本结构用于包含通用的元数据，称为meta box。

### 8.11.1 The Meta box

#### 8.11.1.1 Definition

Box Type: ‘meta’

Container: File, Movie Box (‘moov’), Track Box (‘trak’), or Additional Metadata Container Box (‘meco’)

Mandatory: No

Quantity: Zero or one (in File, ‘moov’, and ‘trak’), One or more (in ‘meco’)

一个meta box包含描述性的或注释性的元数据。“meta” box需要包含一个“hdlr” box，指示“meta” box内容的结构或格式。该元数据位于此box中的box内（例如XML框），或者由主项目box标识的项目中。

所有其他包含的box都特定于处理程序box指定的格式。

这里定义的其他box可以被定义为给定格式的可选box或强制性box。如果使用它们，那么它们必须采用这里指定的形式。这些可选box包括一个data-information box，它记录了放置元数据值（例如图片）的其他文件，以及一个item location box，它记录了这些文件中每个项目所在的位置（例如，在同一文件中存储的多个图片的常见情况下）。在每个文件级别、movie级别或track级别上最多可能出现一个meta box，除非它们包含在一个额外的metadata container box(“meco”)中。

如果出现Item Protection Box，那么部分或全部元数据，其中包括可能的主要资源，可能已经受到保护，并且不可读，除非考虑到保护系统。

#### 8.11.1.2 Syntax

aligned(8) class MetaBox (handler\_type)

extends FullBox(‘meta’, version = 0, 0) {

HandlerBox(handler\_type) theHandler;

PrimaryItemBox primary\_resource; // optional

DataInformationBox file\_locations; // optional

ItemLocationBox item\_locations; // optional

ItemProtectionBox protections; // optional

ItemInfoBox item\_infos; // optional

IPMPControlBox IPMP\_control; // optional

ItemReferenceBox item\_refs; // optional

ItemDataBox item\_data; // optional

Box other\_boxes[]; // optional

}

#### 8.11.1.3 Semantics

元数据的结构或格式由处理程序声明。 在主数据由主项标识的情况下，并且主项有一个带有item\_type的项信息条目，处理程序类型可能与item\_type相同。

### 8.11.2 XML Boxes

#### 8.11.2.1 Definition

Box Type: ‘xml ‘ or ‘bxml’

Container: Meta box (‘meta’)

Mandatory: No

Quantity: Zero or one

当主数据是XML格式的，并且希望将XML直接存储在meta box中时，就可以使用其中一种形式。只有当处理程序标识的已定义格式的XML只有一个定义良好的二值化时，才能使用二进制XMLBox。

在XML box中，数据是UTF-8格式的，除非数据以字节顺序标记(BOM)开头，这表示数据是UTF-16格式的。

#### 8.11.2.2 Syntax

aligned(8) class XMLBox

extends FullBox(‘xml ’, version = 0, 0) {

string xml;

}

aligned(8) class BinaryXMLBox

extends FullBox(‘bxml’, version = 0, 0) {

unsigned int(8) data[]; // to end of box

}

### 8.11.3 The Item Location Box

#### 8.11.3.1 Definition

Box Type: ‘iloc’

Container: Meta box (‘meta’)

Mandatory: No

Quantity: Zero or one

item location box通过定位其包含的文件、它们在该文件中的偏移量和长度，提供了此文件或其他文件中的资源目录。将其放置为二进制格式可以实现对此数据的通用处理，即使是由那些不理解所使用的特定元数据系统（处理程序）的系统也是如此。例如，系统可以将所有外部引用的元数据资源集成到一个文件中，并相应地重新调整文件偏移量和文件引用。

该box以三个或四个值开始，分别指定偏移字段、长度字段、base\_offset字段的字节大小，以及在此box的版本1中指定extent\_index字段的字节大小。这些值必须来自于集合{0、4、8}。

construction\_method字段表示该项目的“构造方法”：

1. file\_offset：通过通常的绝对文件偏移到文件在data\_reference\_index；(construction\_method==0)
2. idat\_offset：通过box偏移到同一meta box中的idat框；既不使用data\_reference\_index或extent\_index字段；(construction\_method==1)
3. item\_offset：通过项目偏移到一个新的extent\_index字段所指示的项目中，该字段仅由此构造方法使用（当前）。(construction\_method == 2)

extent\_index仅用于方法item\_offset；它表示项目引用的基于1的索引，其中包含从该项目链接的引用类型为‘iloc’。如果index\_size为0，则隐含值为1；保留值为0。

项目可以被碎片化存储成区段，例如，以启用交错。扩展区是资源字节的连续子集；资源是通过连接扩展而形成的。如果只使用一个范围(extent\_count=1)，则可以暗示偏移和长度的一个或两个：

* 如果未标识偏移量（字段的长度为零），则暗示源的开头（文件中的偏移量为0、idat框或其他项目）。
* 如果没有指定长度，或指定为零，则暗示了源的整个长度。与此元数据对同一文件的引用，或划分为多个范围的项，应该有显式的偏移量和长度，或者使用一个MIME类型，该类型需要对文件进行不同解释，以避免无限递归。

项目的大小是扩展长度的和。

注意：扩展项可以与track示例表定义的块交错。

数据引用索引可以取值0，它表示与此元数据相同的文件中的引用，或表示数据引用表中的索引。

一些引用的数据本身可能使用偏移/长度技术来处理其中的资源(例如，MP4文件可能以这种方式“包含”)。通常，这种偏移量是相对于所包含的文件的开头出现的。字段“base offse”为包含数据中的偏移量计算提供了额外的偏移量。例如，如果MP4文件包含在此规范格式化的文件中，那么该MP4部分中的数据偏移量通常相对于文件的开头；基本偏移量增加这些偏移量。

如果一个项目是由其他项目构造的，并且这些源项目受到保护，那么偏移量和长度信息将应用于源项目。也就是说，目标项数据由未受保护的源数据形成。

为了最大限度地提高兼容性，如果可能，此框的版本0应该优先使用construction\_method==0的版本1。

#### 8.11.3.2 Syntax

aligned(8) class ItemLocationBox extends FullBox(‘iloc’, version, 0) {

unsigned int(4) offset\_size;

unsigned int(4) length\_size;

unsigned int(4) base\_offset\_size;

if (version == 1)

unsigned int(4) index\_size;

else

unsigned int(4) reserved;

unsigned int(16) item\_count;

for (i=0; i<item\_count; i++) {

unsigned int(16) item\_ID;

if (version == 1) {

unsigned int(12) reserved = 0;

unsigned int(4) construction\_method;

}

unsigned int(16) data\_reference\_index;

unsigned int(base\_offset\_size\*8) base\_offset;

unsigned int(16) extent\_count;

for (j=0; j<extent\_count; j++) {

if ((version == 1) && (index\_size > 0)) {

unsigned int(index\_size\*8) extent\_index;

}

unsigned int(offset\_size\*8) extent\_offset;

unsigned int(length\_size\*8) extent\_length;

}

}

}

#### 8.11.3.3 Semantics

offset\_size取自集合{0、4、8}，并表示offset字段的字节长度。

length\_size取自集合{0、4、8}，以length字段的字节长度表示长度。

base\_offset\_size取自集合{0、4、8}，并表示base\_offset字段的字节长度。

index\_size取自集合{0、4、8}，并表示extent\_index字段的字节长度。

item\_count将计算之后的数组中的资源的数量。

item\_ID是该资源的一个任意整数“名称”，可用于引用它(例如，在URL中)。

construction\_method取自集合0（文件）、1(idat)或2（项）

data-reference-index可以为零（“此文件”），或者是位于数据信息框中的数据引用中的基于1的索引。

base\_offset为引用数据中的偏移量计算提供了一个基本值。如果base\_offset\_size为0，则base\_offset取值0，即它未使用。

extent\_count提供资源被分割到的范围数的计数；它的值必须为1或更大

extent\_index提供了一个为构造方法定义的索引

extent\_offset提供以包含该项文件开头的字节为单位的绝对偏移量。如果offset\_size为0，则offset取值为0

extent\_length提供了此元数据项的字节数的绝对长度。如果length\_size为0，则length取值为0。如果该值为0，则该项的长度是整个被引用文件的长度。

### 8.11.4 Primary Item Box

#### 8.11.4.1 Definition

Box Type: ‘pitm’

Container: Meta box (‘meta’)

Mandatory: No

Quantity: Zero or one

对于给定的处理程序，当希望将主数据存储在其他地方时，它可以是引用项之一，或者划分为范围；或者主元数据可以包含在meta box中（例如在XMLbox中）。必须出现此box，或者在元框中必须有一个box(例如，一个XML box)，其中包含已标识的处理程序所需的格式的主信息。

#### 8.11.4.2 Syntax

aligned(8) class PrimaryItemBox

extends FullBox(‘pitm’, version = 0, 0) {

unsigned int(16) item\_ID;

}

#### 8.11.4.3 Semantics

item\_ID是主项的标识符

8.11.5 Item Protection Box

8.11.5.1 Definition

Box Type: ‘ipro’

Container: Meta box (‘meta’)

Mandatory: No

Quantity: Zero or one

Item protection box提供了一个项目保护信息数组，供Item Information Box使用。

#### 8.11.5.2 Syntax

aligned(8) class ItemProtectionBox

extends FullBox(‘ipro’, version = 0, 0) {

unsigned int(16) protection\_count;

for (i=1; i<=protection\_count; i++) {

ProtectionSchemeInfoBox protection\_information;

}

}

8.11.6 Item Information Box

#### 8.11.6.1 Definition

Box Type: ‘iinf’

Container: Meta Box (‘meta’)

Mandatory: No

Quantity: Zero or one

Item information box提供了有关所选项目的额外信息，包括符号（“文件”）名称。它可以选择性地发生，但如果它发生了，它必须被解释，因为项目保护或内容编码可能已经改变了项目中数据的格式。如果内容编码和保护，读取器应该首先解除保护，然后解码项目的内容编码。如果需要更多的控制，则可以使用IPMP序列代码。

这个box包含一个条目数组，每个条目都被格式化为一个box。这个数组是通过增加输入记录中的item\_ID来排序的。

已定义了项目信息条目的三个版本。版本1包含了由扩展类型指定的版本0的附加信息。例如，对于file partition box (“fpar”)所引用的项目，它应与扩展类型“fdel”一起使用，该文件分区框是为源文件分区定义的，并适用于文件传递传输。版本2提供了一种替代结构，其中元数据项类型由32位（通常是4个字符）的注册或定义的代码表示；其中两个代码表示MIME类型或由URI类型的元数据。

如果不需要扩展，box可以终止没有extension\_type字段和扩展；此外，如果不需要content\_encoding，该字段也可能省略，而box在它之前终止。如果需要一个没有显式content\_encoding的扩展，则必须在指示extension\_type之前，为content\_encoding提供一个表示空字符串的单个空字节。

如果需要文件传递项目信息，并且使用版本2项信息，则文件传递信息作为(a)存储为一个单独的项目“fdel”)(b)，由项目引用从项目链接到“fdel”类型的文件传递信息。如果需要文件传递信息，则必须恰好有一个这样的引用。

MPEG-7元数据可能存在有效的URI形式(例如，带有标识特定元素的片段的模式URI)，而且这些结构也可能用于MPEG-7。然而，在ISO基本媒体文件格式的家族文件中有显式支持MPEG-7，这种显式支持是首选的，因为它允许，除了其他事情：

a)增量更新的元数据(逻辑上，I/P编码，在视频术语中)，而这个草案仅限“I-frame”；

b)二值化和压缩；

c)使用多个模式。

因此，对MPEG-7使用这些结构是不赞成的（而且没有文档记录）。

关于一些元数据系统的URI表单的信息见附录G。

**8.11.6.2 Syntax**

aligned(8) class ItemInfoExtension(unsigned int(32) extension\_type)

{

}

aligned(8) class FDItemInfoExtension() extends ItemInfoExtension (’fdel’)

{

string content\_location;

string content\_MD5;

unsigned int(64) content\_length;

unsigned int(64) transfer\_length;

unsigned int(8) entry\_count;

for (i=1; i <= entry\_count; i++)

unsigned int(32) group\_id;

}

aligned(8) class ItemInfoEntry

extends FullBox(‘infe’, version, 0) {

if ((version == 0) || (version == 1)) {

unsigned int(16) item\_ID;

unsigned int(16) item\_protection\_index

string item\_name;

string content\_type;

string content\_encoding; //optional

}

if (version == 1) {

unsigned int(32) extension\_type; //optional

ItemInfoExtension(extension\_type); //optional

}

if (version == 2) {

unsigned int(16) item\_ID;

unsigned int(16) item\_protection\_index;

unsigned int(32) item\_type;

string item\_name;

if (item\_type==’mime’) {

string content\_type;

string content\_encoding; //optional

} else if (item\_type == ‘uri ‘) {

string item\_uri\_type;

}

}

}

aligned(8) class ItemInfoBox

extends FullBox(‘iinf’, version = 0, 0) {

unsigned int(16) entry\_count;

ItemInfoEntry[ entry\_count ] item\_infos;

}

#### 8.11.6.3 Semantics

item\_id包含主资源的0(例如，包含在“xml”框中的XML)或为其定义以下信息的项的ID。

item\_protection\_index包含未受保护项目的0，或包含item protection box中的定义应用于此项目的保护（item protection box中的第一个box包含索引1）。

item\_name是一个以UTF-8个字符结尾的空结尾字符串，其中包含该项的符号名称（用于文件传递传输的源文件）。

item\_type是一个32位的值，通常是4个可打印的字符，它是一个已定义的有效的项目类型指示符，如“mime”

content\_type是一个以空结尾的以UTF-8编码的字符串，其中的项目具有MIME类型。如果项目是内容编码的（见下文），则内容类型是指内容解码后的项目。

item\_uri\_type是一个具有绝对URI特性的字符串，可用作类型指示符。

content\_encoding是一个可选的以空结尾 UTF-8字符串，用于表示二进制文件已被编码，需要在解释之前进行解码。这些值与HTTP/1.1的内容编码定义一致。一些可能的值是“gzip”、“compress”和“deflate”。空字符串表示没有内容编码。请注意，该项是在应用了内容编码之后才被存储的。

extension\_type是一个可打印的四字符代码，用于标识版本1的版本0的扩展字段。

content\_location是一个UTF-8编码的以空结尾的字符串，其中包含在HTTP/1.1(RFC2616)中定义的文件的URI。

content\_MD5是一个以空结尾的字符串，包含UTF-8字符，其中包含文件的MD5摘要。请参见HTTP/1.1(RFC2616)和RFC1864。

content\_length给出了（未编码的）文件的总长度（以字节为单位）。

transfer\_length给出了（已编码的）文件的总长度（以字节为单位）。请注意，如果不应用内容编码，则传输长度将等于内容长度（见上文）。

entry\_count提供以下数组中条目的计数。

group\_ID表示该文件项（源文件）所属的文件组。有关文件组的更多细节，请参见3GPP TS 26.346。

### 8.11.7 Additional Metadata Container Box

#### 8.11.7.1 Definition

Box Type: ‘meco’

Container: File, Movie Box (‘moov’), or Track Box (‘trak’)

Mandatory: No

Quantity: Zero or one

additional metadata container box包括一个或多个元框。它可以在文件的顶层，在Movie Box(“moov”)，或在track box(“trak”)里面，只有在同一容器中有一个meta box时，它才会出现。不包含在additional metadata container box中的meta box是首选的（主要）meta box。additional metadata container box中的meta box可以补充或提供其他元数据信息。例如，当单个处理程序不能处理所有meta box时，可能需要使用多个meta box。在特定级别上的所有meta box，包括首选的meta box和additional metadata container box中包含的meta box，都必须具有不同的处理程序类型。

包含在additional metadata container box中的meta box应包含primary Item box或处理程序要求的primary data box（例如，XML box）。它不得包括与当前primary Item box或XML box所指示的主项项目相关的box或语法元素。包含在additional metadata container box 中的meta box中的url相对于首选meta box的上下文。

#### 8.11.7.2 Syntax

aligned(8) class AdditionalMetadataContainerBox extends Box('meco') {

}

8.11.8 Metabox Relation Box

#### 8.11.8.1 Definition

Box Type: ‘mere’

Container: Additional Metadata Container Box (‘meco’)

Mandatory: No

Quantity: Zero or more

metabox relation box表示在同一级别上的两个meta box之间的关系，即文件的顶层movie box或track box。如果两个meta box没有metabox relation box，那么这两个meta box之间的关系是未指定的。通过指定其处理程序类型来引用meta boxs。

#### 8.11.8.2 Syntax

aligned(8) class MetaboxRelationBox

extends FullBox('mere', version=0, 0) {

unsigned int(32) first\_metabox\_handler\_type;

unsigned int(32) second\_metabox\_handler\_type;

unsigned int(8) metabox\_relation;

}

#### 8.11.8.3 Semantics

first\_metabox\_handler\_type表示第一个相关的meta box。

second\_metabox\_handler\_type表示第二个相关的meta box。

metabox\_relation表示这两个meta box之间的关系。已定义了以下值：

box之间的关系未知（不存在此box时，这是默认情况）；

1. 这两个box在语义上是不相关的（例如，一个是presentaion，另一个是annotation）；
2. 这两个box在语义上是相关的，但又是互补的（例如，在两个不同的元数据系统中表达的两组不相交的元数据集）；
3. 这两个box在语义上是相关的，但是是重叠的（例如，两组元数据，它们都不是另一组的子集）；两者都不“首选”；
4. 这两个box在语义上是相关的，但第二个box是第一个box的一个适当的子集或较弱的版本；第一个是首选；
5. 这两个box在语义上是相关的和等价的（例如，在两个不同的元数据系统中表达的两组本质上相同的元数据集）。

8.11.9 URL Forms for meta boxes

当使用meta box时，URL可以用来引用meta box中的项目，或使用绝对URL，或使用相对URL。绝对url只能用于引用文件级meta box中的项。

在解释meta box上下文中的数据时（即文件级meta box的文件、movie级meta box的表示或跟踪级meta box的跟踪）时，meta box中的项目被视为与容器文件来自的项目位于相同位置中的shadowing文件。这种shadowing意味着，对与容器文件位于相同位置的另一个文件的引用可以被解析为容器文件本身中的一个项。通过将一个片段附加到容器文件本身的URL中，可以在容器文件中处理项目。该片段以“#”字符开头，由以下两个部分组成：

1. item\_ID=<n>，通过其ID标识项目(主资源的ID可以为0)；
2. item\_name=<item\_name>，当使用项目信息框时。

如果必须处理所包含项中的片段，则该片段的初始“#”字符将替换为“\*”

考虑下面的例子：<http://a.com/d/v.qrv#item\_name=tree.html\*branch1>.我们假设v.qrv是一个在文件级别上具有meta box的文件。首先，客户端剥离碎片并从中获取v.qrv a.然后，它检查顶级meta box，并将其中的项添加到目录“d”的缓存中 a.com.然后它将URL重新形成为<http://a.com/d/tree.html#branch1>.注意，片段已提升为完整的文件名，第一个“\*”已转换为“#”。然后，客户端可以在meta box中找到名为tree.html的项，或者从a.com获取tree.html a.然后它在tree.html中找到锚“branch1”。如果在该html中，一个文件使用一个相对的URL被引用，例如，“flower.gif”，那么客户端将使用普通规则将其转换为一个绝对的URL：<http://a.com/d/flower.gif>，它再次检查flower.gif是否是一个命名项（因此跟踪一个这个名称的单独文件），如果不是，从a.com获取flower.gif.

### 8.11.10 Static Metadata

本节定义了ISO文件格式系列中静态（未定时）元数据的存储方式。

读取器对元数据支持通常是可选的，因此对于这里或其他地方定义的格式也是可选的，除非派生规范是强制性的。

#### 8.11.10.1 Simple textual

现有技术支持使用用户数据box形式的简单文本标记，目前只定义了一个标记——版权声明。在以下情况下，允许使用其他元数据：

* 它使用一个注册的box类型或它使用UUID转义（后者现在是允许的）；
* 它使用一个已注册的标签，等效的MPEG-7结构必须作为注册的一部分记录下来。

#### 8.11.10.2 Other forms

当需要其他形式的meta数据时，则可以在文档的适当级别上包含一个如上面定义的“meta”box。如果文档本身主要是一个元数据文档，那么meta box将位于文件级别。如果meta数据注释了整个演示文稿，则meta box位于movie级别；位于track级别的整个流。

#### 8.11.10.3 MPEG-7 metadata

MPEG-7元数据存储在此规范的meta box中。

1. 处理程序类型是“mp7t”的Unicode格式的文本meta数据；
2. 处理程序类型是“mp7b”，用于以BIM格式压缩的二进制元数据。在这种情况下，二进制XMLbox包含配置信息，后面是二值化的XML。
3. 当格式是文本时，在元数据容器“meta”中有另一个box，称为“xml”，它包含文本MPEG-7文档，或者有一个primary item box，标识包含MPEG-7XML的项目。
4. 当格式是二进制时，元数据容器“meta”中有另一个box称为“bxml”，它包含二进制MPEG-7文档，或者一个标识包含MPEG-7二值化XML的项的primary item box。
5. 如果在文件级别使用了MPEG-7 box，那么brand “mp71”应该是文件类型框中兼容brand列表的成员。

### 8.11.11 Item Data Box

#### 8.11.11.1 Definition

Box Type: ‘idat’

Container: Metadata box (‘meta’)

Mandatory: No

Quantity: Zero or one

此box包含使用构造方法的元数据项的数据，这些构造方法指示项的数据范围存储在此box中。

#### 8.11.11.2 Syntax

aligned(8) class ItemDataBox extends Box(‘idat’) {

bit(8) data[];

}

#### 8.11.11.3 Semantics

data 是所包含的元数据

### 8.11.12 Item Reference Box

#### 8.11.12.1 Definition

Box Type: ‘iref’

Container: Metadata box (‘meta’)

Mandatory: No

Quantity: Zero or one

项引用框允许通过类型的引用将一个项目链接到其他项目。特定类型的一个项目的所有引用将收集到单个项类型引用框中，其类型是引用类型，其中具有“来自项目ID”字段，指示链接的项目。链接到的项然后用“to项目ID”的数组表示。然后，将所有这些单项类型的引用box收集到项引用box中。为8.3.3中定义的track引用box定义的引用类型，或其他注册的引用类型。

注意：通过这种设计，可以很容易地找到特定类型或特定项目的所有引用。

#### 8.11.12.2 Syntax

aligned(8) class SingleItemTypeReferenceBox(referenceType) extends

Box(referenceType) {

unsigned int(16) from\_item\_ID;

unsigned int(16) reference\_count;

for (j=0; j<reference\_count; j++) {

unsigned int(16) to\_item\_ID;

}

}

aligned(8) class ItemReferenceBox extends FullBox(‘iref’, version=0, 0) {

SingleItemTypeReferenceBox references[];

}

#### 8.11.12.3 Semantics

reference\_type包含引用类型的指示

from\_item\_id包含引用其他项的项的ID

reference\_count是引用的数量

to\_item\_id包含所引用的项的ID

### 8.11.13辅助视频元数据

用于深度或视差信息的辅助视频track可以携带一个类型为“auvd”（辅助视频描述符）的元数据项；该项的数据正是ISO/IEC23002-3中规定的一个si\_rbsp()。(注意，si\_rbsp()是外部frame的，长度由文件格式的项目位置信息提供)。可能有多个元数据项（例如，一个用于视差信息，一个用于深度，在同一流服务的情况下）。

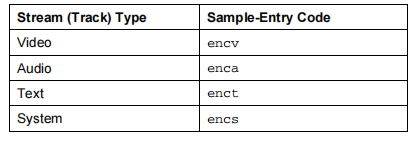
## 8.12 Support for Protected Streams

本节记录了用于受保护内容的文件格式转换。这些转换可以在以下情况下使用：

* 当内容被转换（例如通过加密）时，必须使用它们，以便其不再被普通译码器解码；
* 当只有在理解和实现保护系统时才应该解码内容时，才可以使用它们。

该转换可以通过封装原始媒体声明来发挥作用。封装改变了示例条目的四个字符代码，因此不关心保护的读取器将媒体流视为一种新的流格式。

由于示例条目的格式随媒体类型的不同而不同，因此对每种媒体类型（音频、视频、文本等）使用不同的封装四字符代码。它们分别是：



示例描述的转换过程如下：

1. 示例描述中的四个字符代码被替换为表示保护封装的四个字符代码：这些代码仅因媒体类型而有所不同。例如，“mp4v”被“encv”替换，“mp4a”被“enca”替换。
2. ProtectionSchemeInfoBox（下文定义）添加到示例描述中，不修改所有其他box。
3. 原始示例输入类型（四个字符代码）存储在ProtectionSchemeInfoBox中，并存储在一个名为OriginalFormatBox（下文定义）的新box中；

然后有三种方法来表明保护的性质，它们可以单独使用或联合使用。

1. 当使用MPEG-4系统时，那么必须使用IPMP来发出流被保护的信号。
2. IPMP描述符也可以在MPEG-4系统上下文之外使用包含IPMP描述符的box来使用。
3. 还可以使用方案类型和信息box来描述所应用的保护。

当IPMP在MPEG-4系统之外使用时，一个“global” IPMPControlBox也可能出现在“moov”原子中。

注意：当使用MPEG-4系统时，一个MPEG-4系统终端可以有效地处理，例如，通过使用IPMP描述符，原始格式为“mp4v”的“encv”与“mp4v”完全相同。

### 8.12.1 Protection Scheme Information Box

#### 8.12.1.1 Definition

Box Types: ‘sinf’

Container: Protected Sample Entry, or Item Protection Box (‘ipro’)

Mandatory: Yes

Quantity: Exactly oneOne or More

Protection Scheme Information Box包含了理解所应用的加密转换及其参数所需的所有信息，以及查找密钥管理系统的类型和位置等其他信息所需的所有信息。它还记录了媒体的原始（未加密）格式。Protection Scheme Information Box是一个容器box。在使用指示受保护的流的代码的示例条目中，它是强制性的。

在受保护的示例条目中使用时，此box必须包含原始格式box，以记录原始格式。必须至少使用以下一种信号传递方法来识别所应用的保护措施：

1. 使用IPMP的MPEG-4系统：当使用MPEG-4系统流中的IPMP描述符时，没有其他box；
2. 方案信令：SchemeTypeBox和SchemeInformationBox，当使用它们时（两者都必须存在，或两者都不存在）。

在一个受保护的示例条目中，必须至少有一个protection scheme information box。当发生多个保护时，它们是对相同保护的相同的、替代的描述。读取器应该选择一个来处理。

#### 8.12.1.2 Syntax

aligned(8) class ProtectionSchemeInfoBox(fmt) extends Box('sinf') {

OriginalFormatBox(fmt) original\_format;

SchemeTypeBox scheme\_type\_box; // optional

SchemeInformationBox info; // optional

}

### 8.12.2 Original Format Box

#### 8.12.2.1 Definition

Box Types: ‘frma’

Container: Protection Scheme Information Box (‘sinf’) or Restricted Scheme Information Box (‘rinf’)

Mandatory: Yes when used in a protected sample entry or in a restricted sample entry

Quantity: Exactly one

原始格式box “frma”包含原始未转换的示例描述的四个字符代码：

#### 8.12.2.2 Syntax

aligned(8) class OriginalFormatBox(codingname) extends Box ('frma') {

unsigned int(32) data\_format = codingname;

// format of decrypted, encoded data (in case of protection)

// or un-transformed sample entry (in case of restriction)

}

#### 8.12.2.3 Semantics

data\_format是原始未转换的示例条目的四个字符代码(例如，如果流包含受保护或受限制的MPEG-4视觉材料，则为“mp4v”)。

### 8.12.3 IPMPInfoBox

(empty sub-clause)

### 8.12.4 IPMP Control Box

(empty sub-clause)

### 8.12.5 Scheme Type Box

#### 8.12.5.1 Definition

Box Types: ‘schm’

Container: Protection Scheme Information Box (‘sinf’), Restricted Scheme Information Box (‘rinf’),

or SRTP Process box (‘srpp‘)

Mandatory: No

Quantity: Zero or one in ‘sinf’, depending on the protection structure; Exactly one in ‘rinf’ and ‘srpp’

Scheme Type Box（“schm”）标识保护或限制方案。

#### 8.12.5.2 Syntax

aligned(8) class SchemeTypeBox extends FullBox('schm', 0, flags) {

unsigned int(32) scheme\_type; // 4CC identifying the scheme

unsigned int(32) scheme\_version; // scheme version

if (flags & 0x000001) {

unsigned int(8) scheme\_uri[]; // browser uri

}

}

#### 8.12.5.3 Semantics

scheme\_type是定义保护或限制方案的代码。

scheme\_version是该方案的版本（用于创建内容）

如果用户的系统上没有安装该方案，scheme\_URI允许选择将用户引导到一个网页。它是一个绝对URI，是以UTF-8字符组成的空结尾字符串。

### 8.12.6 Scheme Information Box

#### 8.12.6.1 Definition

Box Types: ‘schi’

Container: Protection Scheme Information Box (‘sinf’), Restricted Scheme Information Box (‘rinf’),

or SRTP Process box (‘srpp‘)

Mandatory: No

Quantity: Zero or one

Scheme Information Box是一个容器box，它只由所使用的方案来解释。加密或限制系统需要的任何信息都存储在这里。此box的内容是一系列box，其类型和格式由Scheme Information Box中声明的方案定义。

#### 8.12.6.2 Syntax

aligned(8) class SchemeInformationBox extends Box('schi') {

Box scheme\_specific\_data[];

}

## 8.13 File Delivery Format Support

### 8.13.1 Introduction

用于通过ALC/LCT或FLUTE传输的文件被存储在一个顶级的元box(“meta”)中。Item location box(“iloc”)指定容器文件中每个项目的实际存储位置以及每个项目的文件大小。每个项目的文件名、内容类型（MIME类型）等均由item information box的版本1（“iinf”）提供。

预先计算的FEC储备作为附加项目存储在元box中。如果一个源文件被分割为多个源块，那么每个源块的FEC库将作为单独的项存储。FEC库和原始源项目之间的关系被记录在FD item information box（“fiin”）中的partition entry box（“paen”）中。

预先组合的文件存储库作为附加项存储在容器文件中。如果一个源文件被分割为多个源块，那么每个源块将被存储为一个单独的项，称为文件存储器。文件库和原始源项之间的关系将记录在FD item information box（“fiin”）中的partition entry box（“paen”）中。

有关文件传递格式使用的更多细节，请参见第9.2款。

### 8.13.2 FD Item Information Box

#### 8.13.2.1 Definition

Box Type: ‘fiin’

Container: Meta Box (‘meta’)

Mandatory: No

Quantity: Zero or one

FD item information box是可选的，尽管它对于使用FD hint track的文件是必需的。它提供了关于源文件的分区以及如何将FD hint track合并成FD会话的信息。每个分区条目都提供了关于特定文件分区、FEC编码以及相关的文件和FEC存储库的详细信息。如果文件中使用替代FEC编码方案或分区，可以为一个源文件(由其项ID标识)提供多个条目。所有的分区条目都被隐式编号，并且第一个条目的编号为1。

#### 8.13.2.2 Syntax

aligned(8) class PartitionEntry extends Box('paen') {

FilePartitionBox blocks\_and\_symbols;

FECReservoirBox FEC\_symbol\_locations; //optional

FileReservoirBox File\_symbol\_locations; //optional

}

aligned(8) class FDItemInformationBox

extends FullBox('fiin', version = 0, 0) {

unsigned int(16) entry\_count;

PartitionEntry partition\_entries[ entry\_count ];

FDSessionGroupBox session\_info; //optional

GroupIdToNameBox group\_id\_to\_name; //optional

}

#### 8.13.2.3 Semantics

entry\_count提供以下数组中条目的计数。

在归档box的地方描述了box的语义。

8.13.3 File Partition Box

#### 8.13.3.1 Definition

Box Type: ‘fpar’

Container: Partition Entry (‘paen’)

Mandatory: Yes

Quantity: Exactly one

文件分区框标识源文件，并将该文件划分为源块和符号。关于源文件的进一步信息，例如文件名、内容位置和组ID，包含在项目信息框(“iinf)中，其中与源文件的项目ID对应的项目信息条目为版本1，并包含一个文件传递项目信息扩展名(“fdel”)。

#### 8.13.3.2 Syntax

aligned(8) class FilePartitionBox

extends FullBox('fpar', version = 0, 0) {

unsigned int(16) item\_ID;

unsigned int(16) packet\_payload\_size;

unsigned int(8) reserved = 0;

unsigned int(8) FEC\_encoding\_ID;

unsigned int(16) FEC\_instance\_ID;

unsigned int(16) max\_source\_block\_length;

unsigned int(16) encoding\_symbol\_length;

unsigned int(16) max\_number\_of\_encoding\_symbols;

string scheme\_specific\_info;

unsigned int(16) entry\_count;

for (i=1; i <= entry\_count; i++) {

unsigned int(16) block\_count;

unsigned int(32) block\_size;

}

}

#### 8.13.3.3 Semantics

item\_ID引用item location box ('iloc')中应用文件分区的项。

packet\_payload\_size给出了分区算法的目标ALC/LCT或FLUTE包有效负载大小。请注意，UDP包的有效负载更大，因为它们还包含ALC/LCT或FLUTE头。

FEC\_encoding\_ID识别FEC编码方案，并接受IANA注册(参见RFC5052)。注意，i)值0对应“紧凑的无代码FEC方案”，也称为“Null-FEC”(RFC3695)；ii)值1对应“MBMSFEC”(3GPPTS26.346)；iii)对于0到127范围内的值，FEC方案完全指定，而对于128到255范围内的值，FEC方案未指定。

FEC\_instance\_ID提供了用于底层指定的FEC方案的FEC编码器的更具体的标识。对于完全指定的FEC方案，这个值应该设置为零，并且在解析包含0到127范围内的FEC\_encoding\_ID文件时应该忽略。FEC\_instance\_ID受到FEC\_encoding\_ID的控制范围。详见RFC5052。

max\_source\_block\_length给出了每个源块的最大源符号数。

encoding\_symbol\_length给出了一个编码符号的大小（字节单位）。一个项目的所有编码符号都具有相同的长度，除了最后一个符号可能更短。

max\_number\_of\_encoding\_symbols给出了编码符号数相关的编码符号数，例如在RFC5052中定义的FEC编码ID129。对于那些编码符号数不相关的FEC方案，该字段的语义不指定

scheme\_specific\_info是方案特定的对象传输信息(FEC-OTI-Scheme特定的信息)的一个base64编码的空终止字符串。该信息的定义取决于FEC的编码ID。

entry\_count给出了提供源文件分区的(block\_count，block\_size)成对列表中的条目数量。从文件的开头开始，每个条目都指示如何将文件的下一个段划分为源块和源符号。

block\_count表示大小为block\_size的连续源块的数量。

block\_size表示块的大小(单位为字节)。 如果block\_size不是encoding\_symbol\_length符号大小的倍数，则使用Compact No-Code FEC表示最后一个源符号包含没有存储在项目中的填充。 使用MBMS FEC (3GPP TS 26.346)，填充可以扩展到多个符号，但填充的大小不能超过encoding\_symbol\_length。

### 8.13.4 FEC Reservoir Box

#### 8.13.4.1 Definition

Box Type: ‘fecr’

Container: Partition Entry (‘paen’)

Mandatory: No

Quantity: Zero or One

FEC库框将文件分区框（“fpar”）中标识的源文件与作为附加项存储的FEC库关联起来。它包含一个列表，该列表从与源文件的第一个源块相关联的第一个FEC存储库开始，并依次通过源文件的源块继续运行。

#### 8.13.4.2 Syntax

aligned(8) class FECReservoirBox

extends FullBox('fecr', version = 0, 0) {

unsigned int(16) entry\_count;

for (i=1; i <= entry\_count; i++) {

unsigned int(16) item\_ID;

unsigned int(32) symbol\_count;

}

}

#### 8.13.4.3 Semantics

entry\_count给出了以下列表中的条目数。这里的条目计数应该与对应的file partition box中的块的总数相匹配。

item\_ID表示与源块相关联的FEC储层的位置。

symbol\_count表示FEC储蓄库中包含的修复符号的数量。

### 8.13.5 FD Session Group Box

#### 8.13.5.1 Definition

Box Type: ‘segr’

Container: FD Information Box (‘fiin’)

Mandatory: No

Quantity: Zero or One

FD会话组框是可选的，尽管它对于包含多个FD提示跟踪的文件是必需的。它包含一个会话列表，以及属于每个会话的所有文件组和hint track。FD会话同时发送特定FD会话在FD会话组box中列出的所有FD hint track（通道）。

随时只应处理一个会话组。会话组中列出的第一个hint track指定基本通道。如果服务器在会话组之间没有首选项，则默认选择应该是第一个会话组。包含hint track所引用的文件的所有文件组的组id都应包含在文件组的列表中。文件组ID也可以依次转换为文件组名称（使用组ID作为名称box）。

#### 8.13.5.2 Syntax

aligned(8) class FDSessionGroupBox extends Box('segr') {

unsigned int(16) num\_session\_groups;

for(i=0; i < num\_session\_groups; i++) {

unsigned int(8) entry\_count;

for (j=0; j < entry\_count; j++) {

unsigned int(32) group\_ID;

}

unsigned int(16) num\_channels\_in\_session\_group;

for(k=0; k < num\_channels\_in\_session\_group; k++) {

unsigned int(32) hint\_track\_id;

}

}

}

#### 8.13.5.3 Semantics

num\_session\_groups指定会话组的数量。

entry\_count给出了以下列表中包含会话组遵守的所有文件组的条目数量。会话组包含由每个源文件的项信息条目指定的所列文件组中包含的所有文件。请注意，会话组的FDT应该只包含在此结构中列出的那些组。

group\_ID表示会话组符合的文件组。

num\_channels\_in\_session\_groups指定会话组中的通道数。num\_channels\_in\_session\_groups的值应为一个正整数。

hint\_track\_ID指定属于特定会话组的FD hint track的track ID。请注意，一个FD hint trackc对应于一个LCT通道。

8.13.6 Group ID to Name Box

#### 8.13.6.1 Definition

Box Type: ‘gitn’

Container: FD Information Box (‘fiin’)

Mandatory: No

Quantity: Zero or One

Group ID to Name box将文件组名称与item information box(“iinf”)中的版本1项目信息条目中使用的文件组ID关联起来。

#### 8.13.6.2 Syntax

aligned(8) class GroupIdToNameBox

extends FullBox('gitn', version = 0, 0) {

unsigned int(16) entry\_count;

for (i=1; i <= entry\_count; i++) {

unsigned int(32) group\_ID;

string group\_name;

}

}

#### 8.13.6.3 Semantics

entry\_count给出了以下列表中的条目数。

group\_ID表示一个文件组。

group\_name是一个以空结尾的字符串，包含UTF-8字符，包含文件组名。

### 8.13.7 File Reservoir Box

#### 8.13.7.1 Definition

Box Type: ‘fire’

Container: Partition Entry (‘paen’)

Mandatory: No

Quantity: Zero or One

File reservoir box将file partition box（“fpar”）中标识的源文件与作为附加项存储的文件存储库关联起来。它包含一个列表，该列表从与源文件的第一个源块相关联的第一个文件存储库开始，并通过源文件的源块依次继续运行。

#### 8.13.7.2 Syntax

aligned(8) class FileReservoirBox

extends FullBox('fire', version = 0, 0) {

unsigned int(16) entry\_count;

for (i=1; i <= entry\_count; i++) {

unsigned int(16) item\_ID;

unsigned int(32) symbol\_count;

}

}

#### 8.13.7.3 Semantics

entry\_count给出了以下列表中的条目数。这里的条目计数应该与相应的file partition box中的总数或块相匹配。

item\_ID表示与源块相关联的文件库的位置。

symbol\_count表示文件存储库中包含的源符号的数量。

## 8.14 Sub tracks

#### 8.14.1 Introduction

Sub track用于分配部分track交替和切换组的方式（整个）track可以分配给交替和切换组，以表明这些track是否彼此替代，以及在会话期间切换是否有意义。Sub track适用于分层介质，例如SVC和MVC，其中的介质替代品往往与track结构不相称。通过在sub track级别上定义替代组和交换组，可以使用现有的规则来进行这种分层编解码器的媒体选择和交换。总体语法对于所有类型的媒体都是通用的，并且与跟踪级定义向后兼容。Sub track级别交替组和交换机组使用与track级别组相同的编号。这些数字在所有track上都是全局的，因此可以跨track和子track边界定义组。

为了定义子track，需要有特定于媒体的定义。SVC和MVC的定义采用AVC文件格式（ISO/IEC14496-15）进行规定。另一种方法是定义样本组，并使用这里定义的Sub Track Sample Group box将它们映射到sub track。该语法也可以进行扩展，以包括其他特定于媒体的定义。

对于应定义的每个sub track，对应track的User Data box。Sub Track box包含定义并提供有关同一track中的sub track的信息的对象。同一track的Track Selection box已经位于这里。

### 8.14.2 Backward compatibility

默认情况下，为（整个）track分配交换组为0（零），这意味着这些（整个）track没有关于替代和/或交换组的信息。但是，即使将sub track定义设置为0，知道sub track定义的文件读取器也能够找到备用组和交换组的sub track信息。通过这种方式，可以通过在file type box中包含适当的品牌，表明遗留读取器可以使用文件。要求读取器知道sub track信息的文件创建者不应该包括传统brand。

如果track中除sub track以外的所有部分都属于同一备用组或交换组，则也可以应用相同的分配sub track信息的方法。然后可以像往常一样在track上进行总体定义，并在sub track上进行具体的分配。对于没有特定分配的sub track，默认情况下会应用跟踪级别分配。与以前一样，如果文件创建者要求读取器知道sub track信息，那么它不应该包括传统brand（否则，这将表明可以跳过sub track信息）。

### 8.14.3 Sub Track box

#### 8.14.3.1 Definition

Box Type: ‘strk’

Container: User Data box (‘udta’) of the corresponding Track box (‘trak’)

Mandatory: No

Quantity: Zero or more

此box包含定义并提供有关当前track中的sub track的信息的对象。

#### 8.14.3.2 Syntax

aligned(8) class SubTrack extends Box(‘strk’) {

}

### 8.14.4 Sub Track Information box

#### 8.14.4.1 Definition

Box Type: ‘stri’

Container: Sub Track box (‘strk’)

Mandatory: Yes

Quantity: One

#### 8.14.4.2 Syntax

aligned(8) class SubTrackInformation

extends FullBox(‘stri’, version = 0, 0){

template int(16) switch\_group = 0;

template int(16) alternate\_group = 0;

template unsigned int(32) sub\_track\_ID = 0;

unsigned int(32) attribute\_list[]; // to the end of the box

}

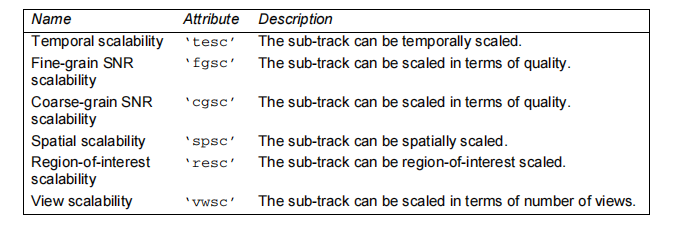
switch\_group是一个整数，它指定track的一组或集合和/或sub track。如果此字段为0（默认值），则没有关于sub track是否可以在播放或streaming期间用于切换的信息。如果此整数不是0，则对于可用于相互切换的track和/或sub track应相同。属于同一交换机组的track应属于同一交换机组。一个交换机组可能只有一个成员。

alternate\_group是一个整数，它指定track的一组或集合和/或sub track。如果此字段为0（默认值），则没有关于与其他track/sub track的可能关系的信息。如果此字段不是0，则对于彼此包含备用数据的track/sub track应该是相同的，而对于属于不同这些组的track/sub track应该是不同的。在任何时候，在一个替代组中，只有一个trackc/sub track应该被播放或播放。

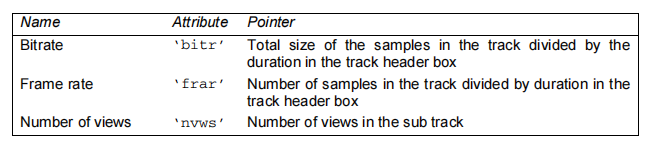
sub\_track\_ID是一个整数。非零值唯一地标识track中本地的sub track。零值（默认值）表示未分配sub track ID。

attribute\_list是一个属性的列表。此列表中的属性应用作sub track的描述，或对同一替代组或开关组中的track和sub track的区分标准。

以下属性是描述性的：



以下属性是有区别的:



### 8.14.5 Sub Track Definition box

**8.14.5.1 Definition**

Box Type: ‘strd’

Container: Sub Track box (‘strk’)

Mandatory: Yes

Quantity: One

此box包含提供sub track定义的对象。

#### 8.14.5.2 Syntax

aligned(8) class SubTrackDefinition extends Box(‘strd’) {

}

### 8.14.6 Sub Track Sample Group box

8.14.6.1 Definition

Box Type: ‘stsg’

Container: Sub Track Definition box (‘strd’)

Mandatory: No

Quantity: Zero or more

此box通过引用描述每个组样本的相应样本组描述，将sub track定义为一个或多个样本组。

#### 8.14.6.2 Syntax

aligned(8) class SubTrackSampleGroupBox

extends FullBox(‘stsg’, 0, 0){

unsigned int(32) grouping\_type;

unsigned int(16) item\_count;

for(i = 0; i< item\_count; i++)

unsigned int(32) group\_description\_index;

}

#### 8.14.6.3 Semantics

grouping\_type是一个标识样本分组的整数。该值应与相应的SampletoGroup和SampleGroupDescription boxes中的值相同。

## 8.15 Post-decoder requirements on media

### 8.15.1 General

为了处理文件作者需要某些动作的情况，这个子条款指定了一个机制，允许播放者简单地检查一个文件找出这样的需求呈现一个位流和停止遗留播放者解码和渲染需要进一步处理的文件。该机制适用于任何类型的视频编解码器。特别是，它适用于AVC，对于这种情况，特定的信令以AVC文件格式(ISO/IEC14496-15)定义，允许文件作者列出正在发生的SEI消息id，并区分呈现过程中必需的和非必需的操作。

该机制类似于内容保护转换，其中示例条目隐藏在通用示例条目：“encv”、“enca”等后面，表示加密或封装的媒体。限制性视频的类似机制使用了一个带有通用示例条目“resv”的转换。当内容只能由正确呈现内容的播放者解码时，就可以应用该方法。

### 8.15.2 Transformation

所应用的方法如下：

1. 示例条目的四个字符代码被一个新的示例条目代码“resv”所取代，意思是限制性视频。
2. 将Restricted Scheme Info box 添加到示例说明中，保留所有未修改的其他box。
3. 原始示例输入类型存储在Restricted Scheme Info box中包含的Original Format box中。

受RestrictedSchemeInfoBox的格式与ProtectionSchemeInfoBox完全相同，只是使用标识符“rinf”而不是“sinf”（见下文）。

原始样本输入类型包含在Restricted Scheme Info box中的Original Format box中（与加密媒体的Protection Scheme Info box相同）。

该限制的确切性质在SchemeTypeBox中定义，该方案所需的数据存储在SchemeInformationBox中，同样，类似于保护信息。

请注意，限制和保护可以同时应用。转换的顺序遵循示例条目的四个字符代码。例如，如果示例输入类型为“resv”，则撤消上述转换可能会得到示例输入类型“encv”，这表明介质受到保护。

请注意，如果文件作者只希望提供建议信息，而不阻止遗留播放器播放该文件，则受限制的方案信息框可以放置在示例条目中，而不应用四字符代码转换。在这种情况下，无需包含Original Format box。

### 8.15.3 Restricted Scheme Information box

#### 8.15.3.1 Definition

Box Types: ‘rinf’

Container: Restricted Sample Entry or Sample Entry

Mandatory: Yes

Quantity: Exactly one

Restricted Scheme Information Box包含了理解所应用的限制方案及其参数所需的所有信息。它还记录了媒体的原始（未转换的）示例输入类型。Restricted Scheme Information Box是一个容器box。它在使用指示受限流的代码的示例项中是强制性的，比如 “resv”。

当在受限制的示例输入中使用时，此box必须包含original format box，以记录原始示例输入类型和Scheme type box。根据限制方案，可能需要一个Scheme Information box。

#### 8.15.3.2 Syntax

aligned(8) class RestrictedSchemeInfoBox(fmt) extends Box('rinf') {

OriginalFormatBox(fmt) original\_format;

SchemeTypeBox scheme\_type\_box;

SchemeInformationBox info; // optional

}

### 8.15.4 Scheme for stereoscopic video arrangements

#### 8.15.4.1 General

当立体编码的视频帧被解码时，解码的帧要么包含两个空间排列的组成帧的表示，要么只包含立体对的一个视图（不同track上的左和右视图）。由于立体声编码的视频而造成的限制包含在Stereo Video box中。

使用了模式类型“stvi”（立体视频）。

#### 8.15.4.2 Stereo video box

##### 8.15.4.2.1 Definition

Box Type: `stvi’

Container: Scheme Information box (‘schi’)

Mandatory: Yes (when the SchemeType is ‘stvi’)

Quantity: One

Stereo Video box用于表示解码帧要么包含构成立体对的两个空间包装的组成帧的表示，要么包含立体对的两个视图中的一个。当模式类型为“stvi”时，应提供Stereo Video box。

##### 8.15.4.2.2 Syntax

aligned(8) class StereoVideoBox extends extends FullBox(‘stvi’, version = 0, 0)

{

template unsigned int(30) reserved = 0;

unsigned int(2) single\_view\_allowed;

unsigned int(32) stereo\_scheme;

unsigned int(32) length;

unsigned int(8)[length] stereo\_indication\_type;

Box[] any\_box; // optional

}

##### 8.15.4.2.3 Semantics

single\_view\_allowed是一个整数。零值表示该内容只能在立体显示器上显示。当(single\_view\_allowed&1)等于1时，允许在单视镜单视图显示器上显示右视图。当(single\_view\_allowed&2)等于2时，允许在单一视角的单视图显示器上显示左视图。

stereo\_scheme是一个整数，指示所使用的立体布置方案和所使用方案的立体指示类型。指定了以下stereo\_scheme的值：

1. ISO/IEC/14496-10[ISO/IEC14496-10]的补充增强信息信息规定的框架包装方案
2. ISO/IEC13818-2[ISO/IEC1381818-2：2000/Amd.4]附件L中规定的布置类型方案。
3. ISO/IEC23000-11中规定的立体声方案，可兼容框架/服务和2D/3D混合服务。

另外，还保留了stereo\_scheme的其他值。

length表示stereo\_indication\_type字段的字节数。

stereo\_indication\_type根据所使用的立体指示方案表示立体布置类型。stereo\_indication\_type的语法和语义依赖于stereo\_scheme的值。以下stereo\_scheme值的stereo\_indication\_type的语法和语义指定如下：

stereo\_scheme等于1：长度值为4，stereo\_indication\_type为无符号int（32），其中包含ISO/IEC14496-10[ISO/IEC14496-10]表D-8中的frame\_packing\_arrangement\_type值（“frame\_packing\_arrangement\_type的定义”）。

stereo\_scheme等于2：长度值为4，stereo\_indication\_type为无符号的int（32），其中包含ISO/IEC13818-2[ISO/IEC1381818-2：2000/Amd4]表L-1中的类型值。(“arrangement\_type的定义”)。

stereo\_scheme等于3：长度的值应为2，stereo\_indication\_type应包含两个无符号int（8）的语法元素。第一个语法元素应包含ISO/IEC23000-11：2009的表4中的立体组合类型。第二个语法元素的最小显著位应包含ISO/IEC23000-11：2009中8.4.3中规定的is\_left\_first值，而其他位应保留，并设置为0。

使用立体声视频框时，适用：

在Track Header box 中

宽度和高度指定拆箱后单个视图的视觉显示大小。

在Sample Description box

frame\_count应该为1，因为解码器在物理上输出一帧。换句话说，frame\_count没有记录帧打包图片中包含的组成帧。

宽度和高度会记录帧填充图片的像素计数（而不是帧填充图片中单个视图的像素计数）。

当视图显示在单视图(monoscopic single-view)显示器上时，Pixel Aspect Ratio box将记录每个视图的像素高宽比。例如，在许多空间帧填充安排中Pixel Aspect Ratio box 因此表示2：1或1：2的像素高宽比，因为与同一格式的帧填充视频的单视图视频相比，其空间分辨率通常沿着一个坐标轴减半。

## 8.16 Segments

### 8.16.1 Introduction

媒体演示可以分为片段进行交付，例如（例如在HTTP流中）形成包含片段或连接片段的文件，这不一定形成ISO基本媒体文件格式兼容的文件（例如，它们不包含movie box）。

本子句定义了可以在这些段中使用的特定box。

### 8.16.2 Segment Type Box

Box Type: `styp’

Container: File

Mandatory: No

Quantity: Zero or more

如果段存储在单独的文件中(例如在标准HTTP服务器上)，建议这些“段文件”包含一个segment-type box，如果存在，必须首先存在，以便识别这些文件，并声明它们兼容的规范。

段类型具有与“ftyp”box[4.3]具有相同的格式，只是它采用了box类型“styp”。其中的brand可能包含在“moov” box之前的“ftyp”box中的相同brand，也可能包括额外的brand，以表明该部分与各种规格的兼容性。

有效的segment type boxes 应为分段中的第一个box。如果段被连接（例如形成完整文件），可以删除segment type boxes ，但这不是必需的。文件中不第一个的segment type boxes 可能被忽略。

### 8.16.3 Segment Index Box

#### 8.16.3.1 Definition

Box Type: `sidx’

Container: File

Mandatory: No

Quantity: Zero or more

Segment Index box（“sidx”）提供了其应用到的媒体段中的一个媒体流的紧凑索引。它的设计不仅可以用于基于本规范的媒体格式（例如包含示例表或movie片段的片段），还可以用于其他媒体格式（例如MPEG-2传输流[ISO/IEC13818-1]）。因此，这里给出的box的形式描述有意描述成通用的，然后在这个子句的末尾给出了使用movie片段的片段的特定定义。

每个分段索引框记录了一个（子）分段如何划分为一个或多个子分段（其本身可以使用Segment Index box进一步进行细分）。

子段被定义为包含（子）段的时间间隔，并对应于包含（子）段的一个字节范围。所有子段的持续时间总和等于包含（子）段的持续时间。

Segment Index box中的每个条目包含一个引用类型，该类型指示引用是直接指向引用的叶子段的媒体字节，还是Segment Index box，描述引用的子段如何进一步细分；因此，通过记录应用于相同（子）段的其他Segment Index box的时间和字节偏移信息，可以以“hierarchical”或“daisy-chain”或其他形式索引。

每个段索引框提供有关片段的单个媒体流的信息，称为参考流。如果提供，对于给定的媒体流，一个段中的第一个Segment Index box应记录该段中的整个媒体流，并应放在同一媒体流的段中的任何其他Segment Index box之前。

如果段中至少一个媒体流但不是所有媒体流存在段索引，那么通常不是每个访问单元都独立编码的媒体流(如视频)被选择为索引。对于没有段索引的任何媒体流，称为非索引流，与段中的第一个Segment Index box相关联的媒体流在某种意义上作为参考流，它也描述了任何非索引媒体流的子段。

注1可在派生规范中规定。

Segment Index boxes可能直接出现在与索引媒体相同的文件中，或者在某些情况下，可能出现在仅包含索引信息的单独文件中。

Segment Index box包含对该box记录的（子）段的子段的引用序列。被引用的子段在表示时间内是连续的。类似地，Segment Index box所引用的字节在媒体文件和单独的索引段中总是是连续的，如果索引放置在媒体文件中，则在单个文件中总是是连续的。所引用的大小给出了所引用材料中的字节数。

注2媒体段可以由多个相互独立的“顶级”Segment Index box对索引进行索引，每个box索引媒体段内的一个媒体流。在包含多个媒体流的段中，被引用的字节可能包含来自多个流的媒体，即使Segment Index box只为一个媒体流提供时间信息。

在包含Segment Index box的文件中，Segment Index box的锚点是该box之后的第一个字节。如果有两个文件，则媒体文件中的锚点是顶级段的开头（即，如果每个段都存储在一个单独的文件中，则是段文件的开头）。包含媒体的文件(也可以是包含Segment Index box的文件)中的材料从从定位点指定的偏移量开始。 如果有两个文件，索引文件中的材料从锚点开始，即紧跟着Segment index box。

在两个约束(a)中，在时间上，子段是连续的，即，循环中的每个条目都是连续的，并且在给定文件（集成文件、媒体文件或索引侧文件）中，引用的字节是连续的，包括以下可能性：

1. 对Segment Index boxes 的引用可以在其字节计数中立即包括在记录子段的段索引box之后；
2. 在一个集成的文件中，使用first\_offset字段，可以将Segment Index boxes 与它们所引用的媒体分开；
3. 在集成文件中，可以找到靠近其索引介质的子段的Segment Index boxes；
4. 当使用包含段索引的单独文件时，循环条目可能是“混合类型”，有些是索引段中的Segment Index boxes，有些是媒体文件中的媒体子段。

注3配置文件可用于限制段索引的位置，或索引的整体复杂性。

Segment Index boxes记录了参考子段中规定的流接入点(SAP)。附件规定了SAP的特征，如ISAU、ISAP和TSAP，以及SAP类型，它们都在下面的语义中使用。当子段包含SAP时，子段以SAP开始，对于第一个SAP，ISAU是继ISAP之后的第一个访问单元的索引，而ISAP包含在子段中。

对于基于此规范的片段（即，基于movie示例表或movie片段）有如下几点：

* 一个访问单元是一个示例；
* 子片段是一个或多个连续的movie片段的独立集合；自包含的集包含一个或多个的movie片段框，包含相应的媒体数据框(es)，包含Movie Fragment boxes引用的数据的媒体数据框必须遵循该Movie Fragment boxes，并在包含相同track信息的下一个Movie Fragment boxes之前；
* Segment Index boxes应放置在其记录的子片段材料之前，即放置在子片段的记录材料的任何影片片段（“moof”）box之前；
* 流是文件格式的轨迹，流id是跟踪id；
* 如果track\_ID等于reference\_ID的流接入点的子段中包含流接入点，则子段包含流接入点；
* sap的初始化数据由电影盒组成；
* presentation时间在movie时间轴中，也就是说它们是应用任何编辑列表后的合成时间；
* 这个ISAP是一个恰好指向一个顶部box的开始的位置，比如一个movie fragment box “moof”；
* 类型1或类型2的SAP表示为同步样本，或在movie片段中的sample\_is\_non\_sync\_sample表示为0；
* 3类型的SAP被标记为“rap”类型的样本组的成员；
* 类型为4的SAP被标记为类型为“roll”的示例组的成员，其中roll\_distance字段的值大于0。

注4对于类型为5和6的SAP，不支持ISO基本媒体文件格式中的特定信令。

#### 8.16.3.2 Syntax

aligned(8) class SegmentIndexBox extends FullBox(‘sidx’, version, 0) {

unsigned int(32) reference\_ID;

unsigned int(32) timescale;

if (version==0) {

unsigned int(32) earliest\_presentation\_time;

unsigned int(32) first\_offset;

}

else {

unsigned int(64) earliest\_presentation\_time;

unsigned int(64) first\_offset;

}

unsigned int(16) reserved = 0;

unsigned int(16) reference\_count;

for(i=1; i <= reference\_count; i++)

{

bit (1) reference\_type;

unsigned int(31) referenced\_size;

unsigned int(32) subsegment\_duration;

bit(1) starts\_with\_SAP;

unsigned int(3) SAP\_type;

unsigned int(28) SAP\_delta\_time;

}

}

#### 8.16.3.3 Semantics

reference\_ID为参考流提供流ID；如果此Segment Index box从“父”Segment Index box中引用，则reference\_ID的值应与“父”Segment Index box的reference\_ID值相同；

timescale以每秒刻度提供时间和持续时间字段的timescale；建议其与参考流或track的timescale匹配；对于基于此规范的文件，即track的Media Header Box的timescale字段；

earliest\_presentation\_time是第一个子段中参考流中任何访问单元最早的呈现时间，在时间尺度字段中指示的时间尺度中；

first\_offset是以字节为单位的距离，在包含媒体的文件中，从锚点到索引材料的第一个字节的距离；

reference\_count提供了被引用的项目的数量；

reference\_type：当设置为1时，表示引用为segment index box（“sidx”），否则引用为媒体内容（例如，基于此规范的文件，指向movie fragment box）；如果使用单独的索引段，则引用类型为1的条目在索引段中，引用类型为0的条目在媒体文件中；

referenced\_size：从字节表示的第一个字节到下一个引用项的第一个字节的距离，或者在最后一个条目的情况下，引用材料的末尾；

subsegment\_duration：当引用为Segment Index box时，该字段携带该box中的subsegment\_duration字段之和；当引用为一个子段时，该字段携带引用流的任何访问单元的第一个子段（或下一个段的第一个子段，或参考流的结束表示时间）与被引用子段中引用流的任何访问单元的最早表示时间之间的差异；持续时间earliest\_presentation\_time的单位相同；

starts\_with\_SAP指示所引用的子段是否以SAP开始。有关此字段与其他字段结合的详细语义，请参见下表。

SAP\_type表示附录I中指定的SAP类型，或值为0。另外，还保留了其他类型的值。有关此字段与其他字段结合的详细语义，请参见下表。

SAP\_delta\_time：在参考流的参考子段中，按解码顺序表示第一个SAP的TSAP。如果引用的子段不包含SAP，则SAP\_delta\_time保留值为0；否则SAP\_delta\_time是子段最早显示时间与TSAP之间的差（注意，在子段以SAP开头的情况下，此差可能为零）。

**Table 4 — Semantics of SAP and reference type combinations**

|  |  |  |  |
| --- | --- | --- | --- |
| starts\_with\_SAP | SAP\_type | reference\_type | Meaning |
| 0 | 0 | 0 or 1 | 没有提供任何关于sap的信息。 |
| 0 | 1 to 6,  inclusive | 0 (media) | 该子段包含（但可能不能以开头）给定SAP\_type的一个SAP，并且给定SAP\_type的第一个SAP对应于SAP\_delta\_time。 |
| 0 | 1 to 6,  inclusive | 1 (index) | 所有引用的子段都包含最多给定SAP\_type的SAP，这些SAP都不是未知类型的。 |
| 1 | 0 | 0 (media) | 子段以未知类型的SAP开始。 |
| 1 | 0 | 1 (index) | 所有被引用的子段都以一个可能是未知类型的SAP开始 |
| 1 | 1 to 6,  inclusive | 0 (media) | 被引用的子段以给定SAP\_type的SAP开始。 |
| 1 | 1 to 6,  inclusive | 1 (index) | 所有引用的子段都以最多给定SAP\_type的SAP开始，这些SAP都不是未知类型的。 |

### 8.16.4 Subsegment Index Box

#### 8.16.4.1 Definition

Box Type: `ssix’

Container: File

Mandatory: No

Quantity: Zero or more

Subsegment Index box（“ssix”）提供了从级别（由级别分配框指定）到索引子段的字节范围的映射。换句话说，这个box提供了一个紧凑的索引，用于说明子段中的数据如何根据级别排序为部分子段。它使客户端能够通过下载子段中的数据范围来轻松地访问部分子段的数据。

子段中的每个字节都应被分配到一个级别。如果该范围与级别分配中的任何信息无关，则可以使用未包含在级别分配中的任何级别。

每个Segment Index box应该有0个或1个子段索引框，它们只索引叶子段，即只索引子段，而不索引段索引。如果有一个子段索引框，则应作为关联的段索引框之后的下一个框。Sub Segment Index box记录在前面的Segment Index box中显示的子段。

一般来说，由字节范围构造的媒体数据是不完整的，即它不符合整个子段的媒体格式。

对于基于此规范的叶子段（即基于movie示例表和movie片段）：

* 每个级别应恰好分配给一个部分子段，即一个级别的字节范围应是连续的。
* 部分子段的水平应通过增加子段内的数量来分配，即，部分子段的样本可能取决于同一子段中先前部分子段的任何样本，但不反过来。例如，每个部分子段包含具有相同时间水平的样本，并且部分子段在该子段中以不断增加的时间水平顺序出现。
* 当以这种方式访问部分子段时，对于除3之外的任何assignment\_type，最终的媒体数据框可能不完整，即，访问的数据比媒体数据框显示的长度指示要少。媒体数据框的长度可能需要调整，或使用填充物。级别分配框中的padding\_flag指示是否可以用零来替换这个缺失的数据。如果没有，则不存在分配到未访问级别的样本的样本数据，应注意不要尝试处理这些样本。

注assignment\_type等于0(在subsegment index box “leva”中指定)，例如，当帧在子段内按时间排序时，可以使用时间级样本分组(“tele”)；例如，当多视图视频位流的每个视图都包含在单独的track中，并且所有视图的track片段都包含在单个的movie片段中。例如，当音频和视频movie片段（包括各自的Media Data boxes）交错时，可以使用等于3的assignment\_type。第一级可以指定为包含音频movie片段（包括相应的Media Data boxes），而第二级可以指定为同时包含音频和视频movie片段（包括所有Media Data boxes）。

#### 8.16.4.2 Syntax

aligned(8) class SubsegmentIndexBox extends FullBox(‘ssix’, 0, 0) {

unsigned int(32) subsegment\_count;

for( i=1; i <= subsegment\_count; i++)

{

unsigned int(32) ranges\_count;

for ( j=1; j <= range\_count; j++) {

unsigned int(8) level;

unsigned int(24) range\_size;

}

}

}

#### 8.16.4.3 Semantics

subsegment\_count是一个正整数，指定在此box中指定部分子段信息的子段的数量。subsegment\_count应等于前一片段索引box中的reference\_count（即movie片段引用数）。

range\_count指定将媒体数据分组到的部分子段级别的数量。该值应大于或等于2。

range\_size表示部分子段的大小。

level指定要分配给此部分子段的级别。

### 8.16.5 Producer Reference Time Box

#### 8.16.5.1 Definition

Box Type: `prft’

Container: File

Mandatory: No

Quantity: Zero or more

producer reference time box提供生成movie片段或包含movie片段（如片段）的文件的相对挂钟时间。当这些文件实时生成和消耗时，这可以为客户提供信息，使消费和生产以相同的速率进行，从而避免可能的缓冲区溢出或欠流。

此box与按照位流顺序跟随它的下一个movie fragment box相关。它必须遵循段中的任何段类型或段索引box（如果有），并且出现在以下movie fragment box （它引用的）之前。如果一个段文件包含任何producer reference time boxes，那么第一个它们应该出现在该段中的第一个movie fragment box之前。

该box包含一个在时钟上测量的时间值，其增量速率与UTC同步NTP[RFC5905]时钟相同，使用NTP格式。这与movie片段中的一个track的媒体时间有关。该媒体时间应该在相关的movie片段中该track的时间范围内。

制作人的参考时间最多应该与一个轨道相关联。

#### 8.16.5.2 Syntax

aligned(8) class ProducerReferenceTimeBox extends FullBox(‘prft’, version, 0) {

unsigned int(32) reference\_track\_ID;

unsigned int(64) ntp\_timestamp;

if (version==0) {

unsigned int(32) media\_time;

} else {

unsigned int(64) media\_time;

}

}

#### 8.16.5.3 Semantics

reference\_track\_ID为参考轨道提供了track\_ID。

ntp\_timestamp表示NTP格式的UTC时间。

media\_time与ntp\_timestamp相同的时间，但在用于参考轨道的时间单位中，并在媒体产生时在这个媒体时钟上测量。

注意：在大多数情况下，这个时间戳不会等于参考track相邻段的第一个样本的时间戳，但建议它在包含这个producer reference time box的段的范围内。

# 9 Hint Track Formats

## 9.1 RTP and SRTP Hint Track Format

### 9.1.1 Introduction

RTP是由IETF(RFC3550和3551)定义的实时传输协议，目前被定义为能够携带一组有限的媒体类型（主要是音频和视频）和编码。将MPEG-4基本流封装到RTP中，这两部分都在讨论中。然而，很明显，媒体被打包的方式与RTP中用于其他编解码器的现有技术没有本质上的区别，并得到了该方案的支持。

在标准的RTP中，每个媒体流作为一个单独的RTP流发送；多路复用是通过使用IP的端口级复用来实现的，而不是通过将来自多个流的数据集成到单个RTP会话中。然而，如果使用MPEG，可能需要将多个媒体轨道多路传输到一个RTP轨道中(例如，当在RTP中使用MPEG-2传输，或FlexMux时)。因此，每个hint track都通过track引用绑定到一组媒体trackc上。hint track通过对这个表进行索引以从它们的媒体track中提取数据。对媒体track的hint track引用的引用类型为“hint”。

此设计决定了创建服务器hint track时的数据包大小；因此，在hint track的声明中，我们指示所选的数据包大小。这包含在示例描述中。请注意，每个媒体track都有多个RTP hint track，同时具有不同的可选包大小是有效的。同样地，还提供了RTP时钟的时间尺度。选择服务器hint track的时间尺度通常会匹配媒体track的时间尺度，或者为服务器选择一个合适的值。在某些情况下，RTP的时间尺度是不同的(例如，对于某些MPEG有效负载为90kHz)，这允许这种变化。会话描述（SAP/SDP）信息存储在track中的user-data boxes 中。

RTP hint track不使用合成时间偏移表（“ctts”）。相反，服务器hint trackc的提示过程建立了正确的传输顺序和时间戳，可能使用传输时间偏移量来设置传输时间。

hint内容可能需要使用SRTP进行streaming，通过使用SRTP的hint track格式，在这里定义。SRTP hint trackc的格式与RTP hint track相同，除了：

1. 示例条目名称从“rtp”更改为“srtp”，以向服务器指示需要SRTP；
2. 在示例条目中添加了一个额外的box，可用于指示服务器使用必须应用的即时加密和完整性保护的性质。

### 9.1.2 Sample Description Format

RTP服务器hint track是hint track（媒体处理程序“hint”），在“rtp”的示例描述中具有条目格式：

class RtpHintSampleEntry() extends SampleEntry (‘rtp ‘) {

uint(16) hinttrackversion = 1;

uint(16) highestcompatibleversion = 1;

uint(32) maxpacketsize;

box additionaldata[];

}

hinttrackversion本当前为1；最高兼容版本字段指定此track向后兼容的最古老的版本。

maxpacketsize表示该track将产生的最大包的大小。

附加数据是一组box，如下。

class timescaleentry() extends Box(‘tims’) {

uint(32) timescale;

}

class timeoffset() extends Box(‘tsro’) {

int(32) offset;

}

class sequenceoffset extends Box(‘snro’) {

int(32) offset;

}

需要输入时间尺度。另外两个都是可选的。偏移量覆盖默认的服务器行为，即选择随机偏移量。因此，值为0将导致服务器分别对时间戳或序列号应用偏移量。

当需要进行SRTP处理时，将使用SRTP提示样例条目。

class SrtpHintSampleEntry() extends SampleEntry (‘srtp‘) {

uint(16) hinttrackversion = 1;

uint(16) highestcompatibleversion = 1;

uint(32) maxpacketsize;

box additionaldata[];

}

字段和box被定义为ISO基本媒体文件格式的RtpHintSampleEntry(“rtp”)。但是，SRTP Process Box 应作为additionaldata boxes之一包含在SrtpHintSampleEntry中。

#### 9.1.2.1 SRTP Process box ‘srpp‘:

Box Type: ‘srpp’

Container: SrtpHintSampleEntry

Mandatory: Yes

Quantity: Exactly one

SRTP进程框可以指示服务器说明应该应用哪些SRTP算法。

aligned(8) class SRTPProcessBox extends FullBox(‘srpp’, version, 0) {

unsigned int(32) encryption\_algorithm\_rtp;

unsigned int(32) encryption\_algorithm\_rtcp;

unsigned int(32) integrity\_algorithm\_rtp;

unsigned int(32) integrity\_algorithm\_rtcp;

SchemeTypeBox scheme\_type\_box;

SchemeInformationBox info;

}

Scheme Type Box和Scheme Information Box具有上面为受保护的媒体轨迹定义的语法。它们可以提供应用SRTP所需的参数。Scheme Type Box用于指示流的必要的密钥管理和安全策略，以扩展到SRTPProcessBox提供的已定义的算法指针。密钥管理功能还用于建立SRTP规范的第8.2节中列出的所有必要的SRTP参数。保护方案的确切定义超出了文件格式的范围。

加密和完整性保护的算法由SRTP定义。在这里定义了以下格式标识符。四个空格（2020202020）可以用来表示加密或完整性保护算法的选择是由文件格式之外的过程决定的。

|  |  |
| --- | --- |
| Format | Algorithm |
| $20$20$20$20 | 加密或完整性保护算法的选择由文件格式之外的进程决定 |
| ACM1 | 使用128位密钥的AES进行加密，如SRTP规范第4.1.1节的定义 |
| AF81 | 使用4.1.2节的f8模式的AES加密，见SRTP规范4.1.2节。 |
| ENUL | 使用SRTP规范第4.1.3节中定义的null算法进行加密 |
| SHM2 | 使用带有160位的HMAC-SHA-1密钥进行完整性保护，如SRTP规范第4.2.1节中所定义的。 |
| ANUL | 完整性保护未应用于RTP(但仍应用于RTCP)。注意：这仅对integrity\_algorithm\_rtp有效 |

### 9.1.3 Sample Format

服务器hint track 中的每个示例都将生成一个或多个RTP数据包，其RTP时间戳与hint示例时间相同。因此，由一个样本制作的所有数据包都具有相同的时间戳。但是，规定要求服务器‘warp’实际传输时间，例如进行数据速率平滑。

每个示例包含两个区域：编写数据包的指令，以及发送这些数据包时所需的任何额外数据（例如，媒体数据的加密版本）。请注意，样本的大小是从样本大小表中知道的

aligned(8) class RTPsample {

unsigned int(16) packetcount;

unsigned int(16) reserved;

RTPpacket packets[packetcount];

byte extradata[];

}

#### 9.1.3.1 Packet Entry format

包输入表中的每个包都具有以下结构：

aligned(8) class RTPpacket {

int(32) relative\_time;

// the next fields form initialization for the RTP

// header (16 bits), and the bit positions correspond

bit(2) RTP\_version;

bit(1) P\_bit;

bit(1) X\_bit;

bit(4) CSRC\_count;

bit(1) M\_bit;

bit(7) payload\_type;

unsigned int(16) RTPsequenceseed;

unsigned int(13) reserved = 0;

unsigned int(1) extra\_flag;

unsigned int(1) bframe\_flag;

unsigned int(1) repeat\_flag;

unsigned int(16) entrycount;

if (extra\_flag) {

uint(32) extra\_information\_length;

box extra\_data\_tlv[];

}

dataentry constructors[entrycount];

}

The semantics of the fields for RTP server hint tracks is specified below. RTP reception hint tracks use the

same packet structure. The semantics of the fields when the packet structure is used in an RTP reception hint

track is specified in subclause 9.4.1.4.

下面指定了有关RTP服务器hint track的字段的语义。RTP接收hint tracks使用相同的数据包结构。当在RTP hint tracks中使用数据包结构时，字段的语义将在子条款9.4.1.4中指定。

在服务器hint tracks中，relative\_time字段“warps”远离样本时间的实际传输时间。这允许流量平滑。

以下2个字节完全覆盖RTP头；它们帮助服务器制作RTP头（服务器填写其余字段）。在这2个字节内，字段RTP\_version和CSRC\_count被保留在服务器（传输）hint track中，服务器将填写这些字段。

序列种子是RTP序列号的基础。如果提示跟踪导致发送同一RTP包的多个副本，那么它们的种子值都是相同的。服务器通常会向此值添加一个随机偏移量（但请参见上面的“sequenceoffset”下）。

extra\_flag等于1表示在构造函数之前有额外的信息，以类型-长度-值集的形式表示。

extra\_information\_length表示构造函数之前所有额外信息的字节长度，其中包括额外的information\_length字段的四个字节。构造函数之前的后续box，称为TLVbox，在32位边界上对齐。任何TLVbox的box大小都表示实际使用的字节，而不是填充到32位边界所需的长度。extra\_information\_length的值包括32位边界所需的填充。

rtpoffsetTLV(“rtpo”)给出了一个32位的有符号整数偏移量来放置在数据包中。这使得数据包可以按解码顺序放在hint track中，但是它们在传输的数据包中呈现时间戳的顺序不同。这对于一些MPEG的有效负载是必要的。

bframe\_flag表示一次性的“b帧”。repeat\_flag表示一个“重复包”，它作为前一个包的副本发送。服务器可能希望优化对这些数据包的处理。

9.1.3.2 Constructor format

构造函数有多种形式。每个构造函数是16个字节，以使迭代更容易。第一个字节是一个联合鉴别器：

aligned(8) class RTPconstructor(type) {

unsigned int(8) constructor\_type = type;

}

aligned(8) class RTPnoopconstructor

extends RTPconstructor(0)

{

uint(8) pad[15];

}

aligned(8) class RTPimmediateconstructor

extends RTPconstructor(1)

{

unsigned int(8) count;

unsigned int(8) data[count];

unsigned int(8) pad[14 - count];

}

aligned(8) class RTPsampleconstructor

extends RTPconstructor(2)

{

signed int(8) trackrefindex;

unsigned int(16) length;

unsigned int(32) samplenumber;

unsigned int(32) sampleoffset;

unsigned int(16) bytesperblock = 1;

unsigned int(16) samplesperblock = 1;

}

aligned(8) class RTPsampledescriptionconstructor

extends RTPconstructor(3)

{

signed int(8) trackrefindex;

unsigned int(16) length;

unsigned int(32) sampledescriptionindex;

unsigned int(32) sampledescriptionoffset;

unsigned int(32) reserved;

}

即时模式允许插入特定于有效载荷的头（例如RTPH.261头）。对于“在清除中”中发送媒体的hint trackc，示例条目接着通过提供要复制的样本号、数据偏移量和长度来指定要从媒体track中复制的字节。track引用可以索引到track引用表（严格的正值），命名hint track本身（-1），或唯一关联的媒体track（0）。（因此，值0等于值1。）

字节块和采样块关注压缩音频，使用一个在MP4之前的方案，其中音频框架在文件中不明显。这些字段对于MP4文件的固定值为1。

sampledescription 模式允许通过引用作为RTP数据包的一部分发送样本描述（其中将包含基本流描述符）。该索引是Sample Description Box中的SampleEntry的索引，并且偏移量相对于该SampleEntry的开头。

对于复杂的情况（例如加密或正向错误纠正），转换后的数据将被放在外部字段中，然后使用引用hint track本身的样本模式。

请注意，没有要求连续的数据包从媒体流传输连续的字节。例如，为了符合H.261的rtp标准包装，有时需要在一个包的末端和下一个包的开头发送一个字节（当宏块边界在一个字节内时）。

### 9.1.4 SDP Information

使用RTSP和SDP的流媒体服务器通常使用SDP作为描述格式；并且在SDP信息和RTP流之间存在必要的关系，例如将有效负载ID映射到MIME名称。因此，我们规定让服务器在文件中保留SDP信息的片段，以帮助服务器形成一个完整的SDP描述。请注意，有必需的SDP条目，服务器也应该生成它们。这里的信息只是部分的。

SDP信息被格式化为user-data boxes中的一组boxes，包括movie级别和track级别。movie级SDP box中的文本应该放在任何特定于媒体的行之前(在SDP文件中的第一个“m=”之前)。

#### 9.1.4.1 Movie SDP information

在movie级别，在用户数据（“udta”）box中，可能会出现一个hint information container box：

aligned(8) class moviehintinformation extends box(‘hnti’) {

}

aligned(8) class rtpmoviehintinformation extends box(‘rtp ‘) {

uint(32) descriptionformat = ‘sdp ‘;

char sdptext[];

}

hint information box可能包含多个协议的信息；此处仅定义了RTP。RTP box可能包含各种描述格式的信息；此处仅定义了SDP。根据SDP的要求，sdptext被正确地格式化为一系列行，每个行都由<crlf>终止。

#### 9.1.4.2 Track SDP Information

在track层面，结构是相似的；但是，从示例描述中，我们已经知道这个track是一个RTPhint track。因此，子文件box仅指定了描述格式。

aligned(8) class trackhintinformation extends box(‘hnti’) {

}

aligned(8) class rtptracksdphintinformation extends box(‘sdp ‘) {

char sdptext[];

}

根据SDP的要求，sdptext 被正确地格式化为一系列行，每个行都由<crlf>终止。

### 9.1.5 Statistical Information

除了hint媒体头中的统计信息外，hinter还可以在hint statistics box中的track user-data box中放置额外的数据。这是一个容器box，其中包含各种可能包含的子box。

aligned(8) class hintstatisticsbox extends box(‘hinf’) {

}

aligned(8) class hintBytesSent extends box(‘trpy’) {

uint(64) bytessent; } // total bytes sent, including 12-byte RTP headers

aligned(8) class hintPacketsSent extends box(‘nump’) {

uint(64) packetssent; } // total packets sent

aligned(8) class hintBytesSent extends box(‘tpyl’) {

uint(64) bytessent; } // total bytes sent, not including RTP headers

aligned(8) class hintBytesSent extends box(‘totl’) {

uint(32) bytessent; } // total bytes sent, including 12-byte RTP headers

aligned(8) class hintPacketsSent extends box(‘npck’) {

uint(32) packetssent; } // total packets sent

aligned(8) class hintBytesSent extends box(‘tpay’) {

uint(32) bytessent; } // total bytes sent, not including RTP headers

aligned(8) class hintmaxrate extends box(‘maxr’) { // maximum data rate

uint(32) period; // in milliseconds

uint(32) bytes; } // max bytes sent in any period ‘period’ long

// including RTP headers

aligned(8) class hintmediaBytesSent extends box(‘dmed’) {

uint(64) bytessent; } // total bytes sent from media tracks

aligned(8) class hintimmediateBytesSent extends box(‘dimm’) {

uint(64) bytessent; } // total bytes sent immediate mode

aligned(8) class hintrepeatedBytesSent extends box(‘drep’) {

uint(64) bytessent; } // total bytes in repeated packets

aligned(8) class hintminrelativetime extends box(‘tmin’) {

int(32) time; } // smallest relative transmission time, milliseconds

aligned(8) class hintmaxrelativetime extends box(‘tmax’) {

int(32) time; } // largest relative transmission time, milliseconds

aligned(8) class hintlargestpacket extends box(‘pmax’) {

uint(32) bytes; } // largest packet sent, including RTP header

aligned(8) class hintlongestpacket extends box(‘dmax’) {

uint(32) time; } // longest packet duration, milliseconds

aligned(8) class hintpayloadID extends box(‘payt’) {

uint(32) payloadID; // payload ID used in RTP packets

uint(8) count;

char rtpmap\_string[count]; }

注意，不是所有这些子box可能存在，并且可能有多个“maxr”box，覆盖不同的时期。

## 9.2 ALC/LCT and FLUTE Hint Track Format

### 9.2.1 Introduction

该文件格式支持具有FEC保护的文件的多播/广播传输。要交付的文件以项的形式存储在容器文件中（由文件格式定义），并在meta box中修改了关于文件如何划分为源符号的信息。对于FEC编码的每个源块，可以预先计算其他奇偶校验符号，并作为FEC存储器项存储。分区取决于FEC方案、目标数据包大小和所需的FEC开销。预先组合的源符号可以存储为文件存储库项，以尽量减少容器文件中的重复信息，特别是使用MBMS-FEC。实际的传输由提示跟踪控制，这些提示跟踪包含有助于将源符号和FEC符号封装成数据包的服务器指令。

FD hint tracks已为ALC/LCT（异步分层编码/分层编码传输）和flock（通过单向传输的文件传输）协议设计。LCT为可靠的内容交付和流交付协议提供了传输级支持。ALC是LCT构建块的一个协议实例化，它可以作为对任意二进制对象的大规模可伸缩的可靠多播分发的基本协议。FLUTE建立在ALC/LCT之上，并定义了一个单向传递文件的协议。

FLUTE定义了一个文件传递表(FDT)，它携带与ALC/LCT会话中传递的文件相关联的元数据，并提供了带内传递和FDT更新的机制。相比之下，ALC/LCT依赖于其他方式在带外交付文件元数据，例如，通常在ALC/LCT会话之前交付给客户端的电子服务指南，以及可以在ALC/LCT会话期间发送的更新片段。

文件分区和FEC储层可以独立于FD hint tracks使用，反之亦然。前者有助于设计hint tracks，并允许替代的hint tracks，例如，使用不同的FEC开销，以重用相同的FEC符号。它们还提供了独立访问源符号和附加FEC符号的手段，用于交付后的修复，这些方法可以通过ALC/LCT或FLUTE或通过另一个协议来执行。为了降低服务器遵循提示跟踪指令时的复杂性，提示跟踪直接引用了复制到提示示例中的项或数据的数据范围。

建议服务器为文件的每次重传发送一组不同的FEC符号。

使用meta box作为源文件的容器文件的语法在8.11中定义，分区、文件和FEC存储库在8.13中定义，而FD提示跟踪的语法在9.2中定义。

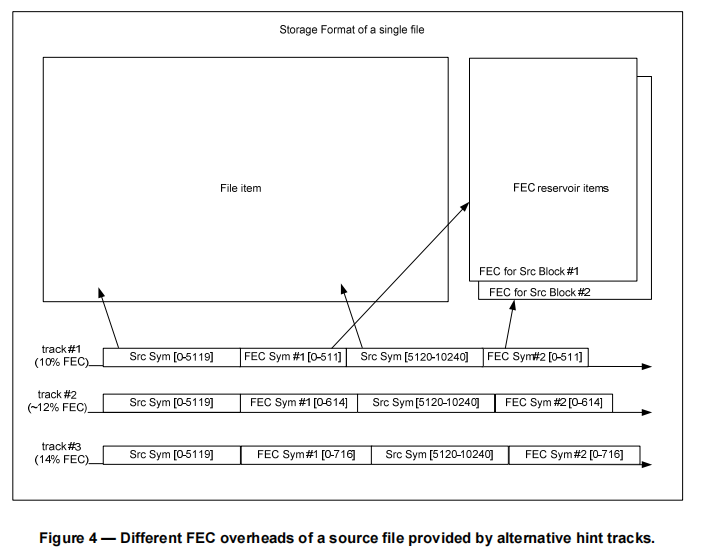
### 9.2.2 Design principles

对文件传递的支持旨在通过使ALC/LCT或FLUTE 服务器能够遵循简单的指令来优化服务器传输过程。对每个通道遵循一个预定义的指令序列来传输一个会话就足够了。文件格式允许存储预先计算的源块和符号分区，即，文件可以被分区为符合预期数据包大小的符号，并预先计算一定数量的fec符号，也可以用于会话后修复。该文件格式还允许存储替代的ALC/LCT或FLUTE 传输会话指令，这可能会导致等效的最终结果。这些替代方案可能适用于不同的信道条件，因为更高的FEC保护，甚至通过使用不同的纠错方案。备选会话可以引用一组公共的符号。

重要的是要区分传输用会话的定义和此类会话的调度。ALC/LCT和FLUTE 服务器文件只处理服务器传输过程的优化。为了确保这些预定义的会话的最大限度的使用和灵活性，所有关于调度地址的细节，等等。被保留在文件格式的定义之外。外部调度应用程序决定了这些细节，而这些细节对于优化传输会话本身并不重要。特别是，以下信息超出了文件格式的范围：时间调度、目标地址和端口、源地址和端口，以及所谓的传输会话标识符(TSI)。

与文件传递hint track的示例相关联的示例编号提供了一个有编号的序列。hint track示例时间提供了针对默认比特率的ALC/LCT或 FLUTE数据包的发送时间。根据实际的传输比特率，ALC/LCT或 FLUTE服务器可以应用线性时间缩放。采样时间可能会简化调度过程，但要由服务器及时发送ALC/LCT或 FLUTE数据包。

图6提供了一个源文件包含三个不同hint tracks的文件的示意图。在本示例中，每个源块只包含一个子块。



上图中的源文件被划分为两个源块，其中包含固定大小的符号。为两个源块计算FEC冗余符号，并作为FEC储存项目存储。当hint track引用文件中相同的项目时，没有重复的信息。原始的源符号和FEC存储库也可以被不使用hint track的修复服务器使用。

### 9.2.3 Sample Description Format

#### 9.2.3.1 Definition

FD hint track是带有handler\_type “hint”和sample description box中的条目格式“fdp”的tracks。FD提示示例项包含在sample description box（“stsd”）中。

#### 9.2.3.2 Syntax

class FDHintSampleEntry() extends SampleEntry ('fdp ') {

unsigned int(16) hinttrackversion = 1;

unsigned int(16) highestcompatibleversion = 1;

unsigned int(16) partition\_entry\_ID;

unsigned int(16) FEC\_overhead;

Box additionaldata[]; //optional

}

#### 9.2.3.3 Semantics

partition\_entry\_ID表示FD item information box中的分区条目。零值表示没有分区条目与此示例条目关联，例如，对于FDT。如果相应的FD hint track只包含开销数据，则此值应该指示其开销数据存在问题的分区条目。

FEC\_overhead是一个固定的8.8值，指示提示示例所使用的保护开销百分比。提供此值的目的是提供特征，以帮助服务器选择会话组（以及相应的FD hint track）。如果相应的FD hint trackc只包含开销数据，则该值应该指示通过使用会话组中直至有问题的FD hint track中的所有FD hint track所实现的保护开销。

后跟踪版本和高度兼容性字段与9.1.2中描述的RTP提示示例条目具有相同的解释。作为附加数据，可以提供一个time scale entry box。如果不提供，则没有关于数据包的定时的指示。

FDT或电子服务指南所需的文件条目可以通过观察提示跟踪的所有示例条目和上述分区条目ID引用的项目所对应的相应item information boxes来创建。如果没有被任何样本引用，任何样本项都不得包含在hint track中。

### 9.2.4 Sample Format

#### 9.2.4.1 Sample Container

hint track中的每个FD样本都将生成一个或多个FD数据包。

每个示例包含两个区域：编写数据包的指令，以及发送这些数据包时所需的任何额外数据(例如，编码复制到示例中的符号，而不是驻留在源文件或FEC的项目中)。请注意，样本的大小是从样本大小表中知道的。

aligned(8) class FDsample extends Box(‘fdsa’) {

FDPacketBox packetbox[]

ExtraDataBox extradata; //optional

}

FD样本的样本数量定义了服务器应处理它们的顺序。同样，每个FD样品中的FD包装箱应按照处理的顺序出现。如果FD hint样本输入中存在 time scale entry box ，则定义采样时间，并为默认比特率提供数据包的相对发送时间。根据实际的传输比特率，服务器可以应用线性时间缩放。采样时间可能会简化调度过程，但要由服务器及时发送数据包。

#### 9.2.4.2 Packet Entry Format

FD示例中的每个数据包都具有以下结构（参考文献：RFC3926、3450、3451）：

aligned(8) class FDpacketBox extends Box(‘fdpa’) {

LCTheaderTemplate LCT\_header\_info;

unsigned int(16) entrycount1;

LCTheaderExtension header\_extension\_constructors[ entrycount1 ];

unsigned int(16) entrycount2;

dataentry packet\_constructors[ entrycount2 ];

}

LCT头信息包含当前FD数据包的LCT头模板。头扩展构造函数是用于构造LCT头扩展的结构。包构造函数用于构造FD包中的FEC有效负载ID和源符号。

#### 9.2.4.3 LCT Header Template Format

LCT报头模板的定义如下：

aligned(8) class LCTheaderTemplate {

unsigned int(1) sender\_current\_time\_present;

unsigned int(1) expected\_residual\_time\_present;

unsigned int(1) session\_close\_bit;

unsigned int(1) object\_close\_bit;

unsigned int(4) reserved;

unsigned int(16) transport\_object\_identifier;

}

服务器可以使用它来形成一个数据包的LCT报头。请注意，报头的某些部分依赖于服务器策略，并且不包含在模板中。一些字段的长度还取决于服务器分配的LCT报头位。服务器可能还需要更改传输对象标识符(TOI)的值。

#### 9.2.4.4 LCT Header Extension Constructor Format

LCT报头扩展构造函数格式定义如下：

aligned(8) class LCTheaderextension {

unsigned int(8) header\_extension\_type;

if (header\_extension\_type > 127) {

unsigned int(8) content[3];

}

else {

unsigned int(8) length;

if (length > 0) {

unsigned int(8) content[(length\*4) - 2];

}

}

长度字段的正值指定构造函数内容的长度，单位为32位字的倍数。零值表示报头是由服务器生成的。

LCT报头扩展的使用和规则在RFC3451(LCTRFC)中定义。header\_extension\_type包含LCT标头扩展类型(HET)值。

介于0到127之间的HET值用于可变长度（多个32位字）扩展。在128到255之间的HET值用于固定长度（一个32位字）的扩展。如果header\_extension\_type小于128，则长度字段对应于RFC3451中定义的LCT报头扩展长度(HEL)。内容字段始终对应于标题扩展内容(HEC)。

注意：服务器可以通过观察是否存在EXT\_FDT(header\_extension\_type==192)来识别包括FDT在内的数据包。

#### 9.2.4.5 Packet Constructor Format

构造函数有多种形式。每个构造函数是16个字节，以便使迭代更简单。第一个字节是一个联合鉴别器。包构造函数用于在FD包中包含FEC有效载荷ID以及源符号和奇偶校验符号。

aligned(8) class FDconstructor(type) {

unsigned int(8) constructor\_type = type;

}

aligned(8) class FDnoopconstructor extends FDconstructor(0)

{

unsigned int(8) pad[15];

}

aligned(8) class FDimmediateconstructor extends FDconstructor(1)

{

unsigned int(8) count;

unsigned int(8) data[count];

unsigned int(8) pad[14 - count];

}

aligned(8) class FDsampleconstructor extends FDconstructor(2)

{

signed int(8) trackrefindex;

unsigned int(16) length;

unsigned int(32) samplenumber;

unsigned int(32) sampleoffset;

unsigned int(16) bytesperblock = 1;

unsigned int(16) samplesperblock = 1;

}

aligned(8) class FDitemconstructor extends FDconstructor(3)

{

unsigned int(16) item\_ID;

unsigned int(16) extent\_index;

unsigned int(64) data\_offset; //offset in byte within extent

unsigned int(24) data\_length; //non-zero length in byte within extent or

//if (data\_length==0) rest of extent

}

aligned(8) class FDxmlboxconstructor extends FDconstructor(4)

{

unsigned int(64) data\_offset; //offset in byte within XMLBox or BinaryXMLBox

unsigned int(32) data\_length;

unsigned int(24) reserved;

}

#### 9.2.4.6 Extra Data Box

FD提示跟踪的每个示例可能包含存储在extra data box中的额外数据：

aligned(8) class ExtraDataBox extends Box(‘extr’) {

FECInformationBox feci;

bit(8) extradata[];

}

9.2.4.7 FEC Information Box

##### 9.2.4.7.1 Definition

Box Type: ‘feci’

Container: Extra Data Box (‘extr’)

Mandatory: No

Quantity: Zero or One

FEC信息框存储在发送FD数据包时所需的FEC编码ID、FEC实例ID和FEC有效载荷ID。

##### 9.2.4.7.2 Syntax

aligned(8) class FECInformationBox extends Box('feci') {

unsigned int(8) FEC\_encoding\_ID;

unsigned int(16) FEC\_instance\_ID;

unsigned int(16) source\_block\_number;

unsigned int(16) encoding\_symbol\_ID;

}

##### 9.2.4.7.3 Semantics

FEC\_encoding\_ID标识FEC编码方案并接受IANA注册（参见RFC5052），其中(i)值零对应于“Compact No-Code FEC scheme”，也称为“Null-FEC”（RFC3695）；(ii)值1对应于“MBMS FEC”（3GPP TS 26.346）；(iii)对于0到127范围的值，包括FEC方案完全指定，而对于128至255范围的值，包括FEC方案未指定。

FEC\_instance\_ID提供了用于底层指定的FEC方案的FEC编码器的更具体的标识。对于完全指定的FEC方案，这个值应该设置为零，并且在解析包含0到127范围内的FEC\_encoding\_ID文件时应该忽略。FEC\_instance\_ID受到FEC\_encoding\_ID的控制范围。详见RFC5052。

source\_block\_number标识FD包中的编码符号来自对象的哪个源块。

encoding\_symbol\_ID标识从源块生成的特定编码符号被携带在FD包中。

## 9.3 MPEG-2 Transport Hint Track Format

### 9.3.1 Introduction

MPEG-2TS（传输流）是一种流多路复用器，它可以携带一个或多个程序，包括音频、视频和其他媒体。该文件格式支持在hint track中存储MPEG-2TS。MPEG-2TS提示跟踪可以用于存储接收到的TS数据包（作为接收hint track），以及作为用于生成MPEG-2 TS的服务器hint track。

MPEG-2 TS hint track定义支持所谓的“预计算hints”。预先计算的hint不使用包括来自其他track的引用数据，而是MPEG-2 TS数据包被这样存储。这允许重用存储在单独文件中的MPEG-2TS数据包。此外，预先计算的hint方便了简单的记录操作。

除了预先计算的hint示例外，还可以通过引用媒体track在hint示例中包含媒体数据。将接收到的传输流转换为媒体track将允许兼容早期版本的ISO基本媒体文件格式的现有播放器处理记录的文件，只要也支持媒体格式。存储原始传输头会保留有价值的信息，用于错误隐藏和重建原始传输流。

### 9.3.2 Design Principles

MPEG-2TS提示轨迹格式的设计原则如下。

MPEG-2 TS hint track中的样本序列是一组预先计算和构造的MPEG-2 TS数据包。预先计算的包是在接收或将被发送的TS包。当数据不能多路复用且不能创建基本流时尤其重要，当传输流被加密并且不允许解密存储时。因此，有必要能够将MPEG-2 TS这样存储在一个hint track中。构造的数据包使用与RTP hint trackc相同的方法，即，示例包含一个流媒体服务器构建数据包的指令。实际的媒体数据包含在其他track中。使用了“huint”类型的track引用。

#### 9.3.2.1 Reusing existing Transport Streams

我们希望重用现有的TS实例，因此存在一个额外的机制来覆盖各种各样的现有TS记录。这些记录不仅包括TS包，而且包含每个TS包的前或尾数据。前面数据的一种特殊情况是在每个TS包前面有一个4字节的时间戳，以消除传输系统的抖动。跟踪数据的一种特殊情况是，当TS数据包在一个容易出错的信道上传输时，会添加FEC。

#### 9.3.2.2 Timing

MPEG-2TS为每个程序定义一个时钟，运行在27MHz，其采样值作为pcr在TS中传输，用于时钟恢复。MPEG-2 TS hint track的时间尺度建议为90000，或采用整数除法或其倍数。

MPEG-2 TS Hint Track中一个样品的解码时间是该包或包组的第一个比特的接收/发送时间，建议从TS的PCR时间戳中推导出来，因为如果使用PCR时间，分段线性可以假定和“stts”表紧凑。示例描述中可选的“tsti” box可用于指示当hint track是reception hint track时，是否使用有时钟恢复的接收定时。在服务器hint track的情况下，假设PCR timming。

注意：当一个样本中有多个数据包时，不能给它们独立的传输时间偏移量。

#### 9.3.2.3 Packet Grouping

MPEG-2传输流hint track的示例格式允许在一个示例中包含多个TS数据包。特定的应用程序，如一些IPTV应用程序，在RTP流中传输TS数据包。对于一个RTP包中携带的所有TS包，只能导出一个接收时间戳。另一个用于在一个示例中存储多个TS数据包的应用程序是SPTSs，其中一个示例包含一个GoP的所有TS数据包。在这种情况下，每个样本都是一个随机访问点。

请注意，如果使用每个样本使用多个TS包，则不可能通过文件格式随机访问每个TS包。

在MPTS的情况下，每个样本只应该使用一个数据包。这便于在每个包的基础上使用样本组机制。

#### 9.3.2.4 Random-access points

同步示例意味着一个处理track可能开始没有错误。MPTS和SPTS都支持MPEG-2 TS hint track，但是一个随机接入点，标记为同步示例，通常只为SPTS定义，它指定一个包的开始，包含使用流的独立可解码媒体访问单元(例如使用差分编码的MPEG-2视频I-帧或MPEG-4 AVC IDR图片)的第一个字节。对于MPTS，同步样本表通常存在，但是空的，这表明在track中没有点，整个track的处理开始可能没有错误。建议将PSI/SI放在示例描述中，以便能够仅使用媒体数据进行真正的随机访问。

请注意，在MPTS的情况下，同步示例表存在，但为空（这意味着基本上没有示例是同步示例）。

还要注意，对于SPTS的情况，包含多个TS数据包的示例在示例的开始时应该有一个同步点(例如，GoP边界)。然后，同步示例表将示例标记为同步点（例如，GoPs的开始）；如果缺少同步示例表，则所有示例都是同步点。如果同步样本表存在但为空，则同步样本位置未知，可能不是在样本开始时。

注意：搜索关键帧的应用程序可以在该位置开始读取，但通常它还必须读取进一步的MPEG-2TS包（关于文件格式，这些是后续的样本），以便解码器可以解码一个完整的帧。

#### 9.3.2.5 Application as a Reception Hint Track

当记录一个或多个数据包流时，可以使用Reception hint tracks 。它们指示订单、接收时间和接收到的数据包的内容等。

注1：播放人员可以根据接收提示轨迹复制接收到的包流，并处理复制的包流，就像处理新接收到的一样。

Reception hint tracks与服务器的hint tracks具有相同的结构。

reception hint样本的格式由Reception hint tracks的样本描述表示。每个协议都有自己的Reception hint 示例格式和名称。

注2：服务器使用reception hint tracks 作为发送接收流的hints，应处理接收流的潜在降级，例如传输延迟抖动和包丢失，优雅地确保遵守协议和包含数据格式的约束，而不管接收流的潜在降级

注3：与server hint tracks一样， reception hint tracks的示例格式可以通过通过引用从其他track中提取数据来构建数据包。这些其他的track可能是hint track或媒体track。这些指针的确切形式由协议的示例格式定义的，但通常它们由四条信息组成：一个track参考索引、一个示例号、一个偏移量和一个长度。其中一些可能是针对特定的协议所隐含的。这些“指针”总是指向数据的实际来源，即，不允许进行间接的数据引用。如果一个hint track是建立在另一个hint track的“顶部”，那么第二个hint track必须直接引用第一个媒体track使用的媒体track，其中来自这些媒体track的数据被放置在流中。

如果将接收到的数据提取到媒体track，删除过程必须确保媒体流有效，即流必须无错误（例如需要错误隐藏）。

在接收提示track中允许有一个大小为零的样本，这样的样本可能会被忽略。

### 9.3.3 Sample Description Format

9.3.3.1 Introduction

MPEG2-TS reception hint track的示例描述包含描述流或其部分的所有静态元数据，特别是PSI/SI表。MPEG-2 TS接收提示跟踪在“rm2t”示例描述（表示MPEG-2传输流）中使用输入格式。MPEG2-TS服务器提示跟踪的输入格式是“sm2t”。

静态元数据文档，例如PSI/SI表。静态元数据的存在是可选的。存在时，静态元数据应对其描述的MPEG2-TS数据包有效。因此，如果流中的一段静态元数据发生更改，则在更改时或之后对第一个示例需要一个新的示例条目。如果示例条目中没有静态元数据，那么存储在MPEG2-TS数据包中的结构，如PSI/SI表，是有效的，并且必须扫描流，以找出静态元数据的哪些值对特定的示例是有效的。

#### 9.3.3.2 Syntax

class MPEG2TSReceptionSampleEntry extends MPEG2TSSampleEntry(`rm2t´) {

}

class MPEG2TSServerSampleEntry extends MPEG2TSSampleEntry(`sm2t´) {

}

class MPEG2TSSampleEntry(name) extends HintSampleEntry(name) {

uint(16) hinttrackversion = 1;

uint(16) highestcompatibleversion = 1;

uint(8) precedingbyteslen;

uint(8) trailingbyteslen;

uint(1) precomputed\_only\_flag;

uint(7) reserved;

box additionaldata[];

}

#### 9.3.3.3 Semantics

hinttrackversion 目前是1；highestcompatibleversion 字段指定了该track向后兼容的最古老的版本。

precedingbyteslen teslen表示每个MPEG2-TS包之前的字节数（例如可以是来自外部记录设备的时间码）。

trailingbyteslen 表示每个MPEG2-TS包末尾的字节数（例如，可能包含校验和或由记录设备添加的校验和或其他数据）。

precomputed\_only\_flag表示如果设置为1，相关的样本是否是纯预先计算的；

additionaldata 是一组盒子。这个集合可以包含通过“tPAT” box或“tPMT” box或其他数据描述PSI/SI表常见版本的box，例如仅适用于样本的box（包含多个包），并描述STC的初始条件或定义前面或后面数据内容的box。附加数据中最多应有一个 PATBox、TSTimingBox、InitialSampleTimeBox、InitialSampleTimeBox

定义了以下additionaldata 的可选boxes：

aligned(8) class PATBox() extends Box(‘tPAT’) {

uint(3) reserved;

uint(13) PID;

uint(8) sectiondata[];

}

aligned(8) class PMTBox() extends Box(‘tPMT’) {

uint(3) reserved;

uint(13) PID;

uint(8) sectiondata[];

}

aligned(8) class ODBox () extends Box (‘tOD ’) {

uint(3) reserved;

uint(13) PID;

uint(8) sectiondata[];

}

aligned(8) class TSTimingBox() extends Box(‘tsti’) {

uint(1) timing\_derivation\_method;

uint(2) reserved;

uint(13) PID;

}

aligned(8) class InitialSampleTimeBox() extends Box(‘istm’) {

uint(32) initialsampletime;

uint(32) reserved;

}

“tPAT”框包含PAT的分段数据，每个“tPMT” box包含其中一个PMT的分段数据。

在SPTS的情况下，强烈建议在additionaldata中出现“tPMT” box。如果样本数据中不存在PMT，则应在additionaldata中存在。如果存在“tPMT”框，它应该是样本数据中包含的程序的PMT(尽管记录的流可能包含其他程序，并且是一个MPTS)。

PID是从中提取数据的MPEG2-TS数据包的PID。在“tPAT”框中，此值始终为0。

sectiondata 扩展到方框的末尾，是完整的MPEG2-TS表，其中包含id版本号的连接部分。

initialsampletime 指定采样时间的初始值，如果采样时间不是从0开始的。与媒体track不同，MPEG-2 TS hint track通常没有从0开始的采样时间，例如，PCR时间和接收时间。由于“stts”只存储采样时间之间的增量，因此此字段需要重建原始采样时间：

OriginalSampleTime(n) = initialsampletime + STTS(n).

如果样本时间使用PCR时间，那么当随机访问样本时，可以使用重构的样本时间来初始化STC。请注意，在编辑后可能需要更新此字段。

timing\_derivation\_method是一个标志，它指定用于为给定的PID设置采样时间的方法。timing\_derivation\_method的值如下：

0x0接收时间：采样时间来自于接收时间。不能保证STC恢复接收时间。

0x1 pcr之间的分段线性：样本时间来自于该程序重建的一个STC。假设相邻pcr之间的分段线性，并且样本中的所有TS包在这个范围内都有一个恒定的持续时间。

### 9.3.4 Sample Format

一个MPEG-2 TS hint track的每个示例都由一组

预先计算的包：一个或多个MPEG-2TS包与相关的报头和报尾。

构造的包：通过指向另一个track的数据，由与相关的报头和报尾组成一个或多个MPEG2-TS包的指令。

请注意，示例中的每个MPEG-2 TS包之前可能带有一个前标题（前字节），或后面有一个后预告片（后字节），详见示例描述格式。在示例描述中，前标题和后标题的大小分别由precedingbyteslen 和trailingbyteslen来指定，以允许具有更少块的紧凑示例表。

预先计算的样本和构造的样本的混合有可能发生在同一track上。如果需要填充传输流包，这可以通过adaptation\_field或适当地使用MPEG2TSImmediateConstructor显式来完成。

注意：如果样本只包含预先计算的包，那么样本中MPEG-2 TS包的数量可以直接从样本大小表中得到，如果设置了示例条目中的precomputed\_only\_flag，则为结论。示例中MPEG-2 TS包的数量可能是可变的或受限制的，例如，该文件格式的扩展名可以定义为只包含一个包。

#### 9.3.4.1 Syntax

// Constructor format

aligned(8) abstract class MPEG2TSConstructor (uint(8) type) {

uint(8) constructor\_type = type;

}

aligned(8) class MPEG2TSImmediateConstructor

extends MPEG2TSConstructor(1) {

uint(8) immediatedatalen;

uint(8) data[immediatedatalen];

}

aligned(8) class MPEG2TSSampleConstructor

extends MPEG2TSConstructor(2) {

uint(8) sampledatalen;

uint(16) trackrefindex;

uint(32) samplenumber;

uint(32) sampleoffset;

}

// Packet format

aligned(8) class MPEG2TSPacketRepresentation {

uint(8) precedingbytes[precedingbyteslen];

uint(8) sync\_byte;

if (sync\_byte == 0x47) {

uint(8) packet[187];

} else if (sync\_byte == 0x00) {

uint(8) headerdatalen;

uint(4) reserved;

uint(4) num\_constructors;

bit(1) transport\_error\_indicator;

bit(1) payload\_unit\_start\_indicator;

bit(1) transport\_priority;

bit(13) PID;

bit(2) transport\_scrambling\_control;

bit(2) adaptation\_field\_control;

bit(4) continuity\_counter;

if (adaptation\_field\_control == ´10´ ||

adaptation\_field\_control == ´11´) {

uint(8) adaptation\_field[headerdatalen-3];

}

MPEG2TSConstructor constructors[num\_constructors];

}

uint(8) trailingbytes[trailingbyteslen];

}

// Sample format

aligned(8) class MPEG2TSSample {

MPEG2TSPacket packet[];

}

#### 9.3.4.2 Semantics

precedingbytes 包含在数据包之前的任何额外数据，通常由记录设备提供。例如，这可能包括一个时间戳。

sync\_byte：如果此值为0x47，那么示例是传输流数据包（预先计算的reception hint track示例），其余字节跟随于字段数据包中。如果此值为0x00，它表示关联的示例指向引用类型为“hint”的track reference box中由trackrefindex索引的轨迹。当前已保留了所有其他值。当数据包数据实际放入流媒体通道时，该值应始终设置为0x47。

trackrefindex track reference box中引用类型为“hint”的索引，用于指示当前样本与哪个媒体track相关联。MPEG2TSSampleConstructor 中的samplenumber 和sampleoffset字段指向这个媒体track。轨迹从值1开始。保留值0以供将来使用。

packet MPEG数据包，除了同步字节（0x47）-2TS数据包。

MPEG2TSConstructor数组是一个或多个构造函数项的集合，允许在一个传输流包中有多个访问单元。 MPEG2TSImmediateConstructor可以包含PES头文件。 MPEG2TSSampleConstructor引用相关的媒体track中的数据。 MPEG2TSPacket的所有构造函数的headerdatalen和datalen字段的总和必须等于正在构造的传输流包的长度，减去1字节，即187。

trailingbytes 包含跟在数据包后面的任何额外数据。例如，这可能包括一个校验和。

samplenumber 表示包中包含的参考track中的样本，sampleoffset表示包含sampledatalen字节的包中包含的引用媒体样本的起始字节位置。 sampleoffset 从值0开始。

Immediatedatalen表示示例中包含的字段数据的字节数，而不是通过引用媒体track包含到示例中的数据。

headerdatalen TS报文头长度(不带同步字节)，单位为字节。 如果adaptation\_field不存在，这个字段的值为3或值(adaptation\_field\_length+3)，其中adaptation\_field\_length是结构adaptation\_field的第一个八位元，在ISO/IEC 13818-1中定义。

该规范既没有定义precedingbytes 的格式，也没有定义 trailingbytes的格式。

示例结构的其余字段(transport\_error\_indicator、payload\_unit\_start\_indicator、transport\_priority、PID、transport\_scrambling\_control、adaptation\_field\_control、continuity\_counter、adaptation\_field)包含TS包的包头副本，如ISO/IEC13818-1中定义的那样。

### 9.3.5 Protected MPEG 2 Transport Stream Hint Track

#### 9.3.5.1 Introduction

本子条款定义了一种将媒体流标记为受保护媒体流的机制。这可以通过更改样本输入的四个字符代码和附加包含保护机制细节和原始四个字符代码的box来实现。但是，在这种情况下，track不受保护；它是一个包含受保护数据的明确hint track。本子句描述了如何将hint track标记为携带受保护的数据，使用类似的机制和使用相同的box。

#### 9.3.5.2 Syntax

class ProtectedMPEG2TransportStreamSampleEntry

extends MPEG2TransportStreamSampleEntry(‘pm2t’) {

ProtectionSchemeInfoBox SchemeInformation;

}

#### 9.3.5.3 Semantics

The SchemeInformation ("sinf") box (defined in 8.12) shall contain details of the protection scheme applied.

This shall include the OriginalFormatBox which shall contain the original sample entry type of the MPEG-2

Transport StreamSampleEntry box.

SchemeInformation （“sinf”）box（定义见8.12）应包含所应用的保护方案的详细信息。这应包括OriginalFormatBox，其中应包含MPEG-2传输StreamSampleEntry 的原始样品入口类型。

## 9.4 RTP, RTCP, SRTP and SRTCP Reception Hint Tracks

### 9.4.1 RTP Reception Hint Track

#### 9.4.1.1 Introduction

此子句指定实时传输协议(RTP)的接收提示跟踪格式，如IETFRFC3550中定义的那样。

RTP用于通过互联网协议进行的实时媒体传输。每个RTP流携带一种媒体类型，而一个RTP接收hint track携带一个RTP流。因此，一个audio-visual程序的记录会导致至少两个RTP接收hint track。

RTP接收hint track格式的设计尽可能遵循RTP服务器hint track格式的设计。这种设计应该确保RTP数据包传输的运行方式非常相似，无论它是基于RTP接收hint track还是RTP服务器hint track。此外，文件格式中的新数据结构的数量因此保持得尽可能少。

RTP接收hint track的格式允许在hint样本中存储数据包有效负载，或者将RTP数据包有效负载转换为媒体样本，并通过引用hint样本来包括它们，或者结合这两种方法。如前所述，将接收到的流转换为媒体track，允许符合ISO基本媒体文件格式早期版本的现有播放器处理录制的文件，只要这种媒体格式也受到支持。存储原始RTP头会保留有价值的信息，以隐藏错误和重建原始RTP流。需要注意的是，在预先计算的RTP接收提示轨道中的流的记录完成后，数据包有效载荷到媒体样本的转换可能会发生“"off-line”。

#### 9.4.1.2 Sample Description Format

RTP接收hint track的示例描述中的条目格式是“rrtp”。示例条目的语法与具有条目格式为“rtp”的RTP服务器hint track相同。

class ReceivedRtpHintSampleEntry() extends SampleEntry (‘rrtp‘) {

uint(16) hinttrackversion = 1;

uint(16) highestcompatibleversion = 1;

uint(32) maxpacketsize;

box additionaldata[];

}

RTP接收hint track的示例描述中的条目格式标识符与RTP服务器hint track的示例描述中的条目格式不同，以避免使用包含错误的RTP接收hint track作为有效的服务器hint track。

box的附加数据集可能包括时间尺度输入（“tims”）和时间偏移量（“tsro”）box。此外，附加数据可能包含一个timestamp synchrony box。

应提供时间尺度输入box（“时间”），且时间尺度的值应设置为匹配在接收hint track中捕获的流的RTP时间戳的时钟频率。

可能存在 time offset box（“tsro”）。如果 time offset box不存在，则推断字段偏移量的值等于0。字段offset的值用于RTP时间戳的推导，如9.4.1.4中所述。

RTP时间戳通常不会从零开始，特别是当RTP接收器“调整”到一个流时。因此，时间偏移框应该出现在RTP接收hint track中，并且time offset box中的偏移值应该被设置为等于接收顺序中的RTP流的第一个RTP时间戳。

在RTP接收hint track的样本条目的附加数据中可能存在零个或一个timestampsynchrony box。如果不存在timestampsynchrony ，则推断timestamp\_sync的值等于0。

class timestampsynchrony() extends Box(‘tssy’) {

unsigned int(6) reserved;

unsigned int(2) timestamp\_sync;

}

timestamp\_sync = 0表示由9.4.1.4公式导出的当前RTP接收提示跟踪的RTP时间戳可能与其他RTP接收hint track的RTP时间戳同步，也可能不同步。

timestamp\_sync等于1表示从9.4.1.4中的公式派生的当前RTP接收hint track的RTP时间戳准确反映接收的RTP时间戳（不修正与任何其他RTP接收提示轨迹的同步）。

timestamp\_sync等于2表示从9.4.1.4中的公式衍生的当前RTP接收hint track的RTP时间戳与其他RTP接收hint track的RTP时间戳同步。

当timestamp\_sync等于0或1时，播放者应该使用存储的RTCP发送方报告来纠正流间的同步。当timestamp\_sync等于2时，RTP接收提示轨道中包含的媒体可以根据重构的RTP时间戳同步播放，而无需使用RTCP发送者报告进行同步校正。如果期望RTP接收hint track将用于重新发送已记录的RTP流，则建议将timestamp\_sync设置为0或1，因为存储的RTCP发送者报告可以被重用。

保留timestamp\_sync等于3。

对于文件中出现的所有RTP接收hint track，timestamp\_sync的值应完全相同。

When RTCP is also stored, using an RTCP hint track, the timestamp relationship between the RTP and RTCP

hint tracks can only be maintained if the RTP timestamps are anchored by using a set time offset (‘tsro’) in the

RTP track, and hence the time offset is mandatory if RTCP is stored in an RTCP hint track

当RTCP也被存储，使用RTCP hint track，RTP和RTCP提示跟踪之间的时间戳关系，只有在RTP时间戳使用set time offset(' tsro ')在RTP track中锚定时才能被维护，因此如果RTCP存储在RTCP hint trackz则时间偏移是必须的。

在RTP接收hint track的示例描述符条目的附加数据中，应包含具有四个字符代码“rssr”的零个或一个接收器dssrcBox：

class ReceivedSsrcBox extends Box(‘rssr’) {

unsigned int(32) SSRC

}

SSRC值必须等于示例描述所描述的所有记录SRTP数据包的头中的SSRC值。

#### 9.4.1.3 Sample Format

RTP接收hint track的示例格式与RTP服务器hint track的示例格式的语法相同。接收hint track中的每个示例代表一个或多个接收到的RTP数据包。如果媒体帧在RTP流中不同时是碎片化和交错的，则建议每个样本代表所有具有相同RTP时间戳的已接收的RTP数据包，即，以RTP序号顺序和一个共同的RTP时间戳的连续数据包。

每个RTP接收提示示例包含两个区域：编写数据包的指令，以及编写数据包所需的任何额外数据，例如数据包有效负载的副本。请注意，样本的大小是从样本大小表中知道的。

由于数据包的接收时间可能会发生变化，因此该变化可以为随后指定的每个包发出信号。

在接收hint track中允许有一个大小为零的样本，这样的样本可能会被忽略。

#### 9.4.1.4 Packet Entry Format

数据包输入表中的每个数据包都具有与服务器（传输）hint track相同的结构，见9.1.3.1。

其中i为样本样本编号，8.6.1.2和relative\_time中规定的样本时间DT(i)之和表示数据包的接收时间。接收时间的时钟源是未定义的，例如可以是接收器的挂钟。如果一个接收hint track的接收时间范围与另一个接收hint track的接收时间范围完全或部分重叠，则这些提示轨道的时钟源应该是相同的。

建议接收方通过在存储的接收hint sample中自适应地设置relative\_time，在sample box(stts)中，使用合理和平滑的包调度和端到端延迟变化。这种设置sample\_delta和relative\_time值的安排可以方便压缩解码时间到sample box。在这种情况下，timestamp\_sync被设置为1，样本持续时间大多是常数，时间偏移量(“tsro”)存储在样本条目中。

RTP\_version、P\_bit、X\_bit、CSRC\_count、M\_bit、payload\_type、RTPsequenceseed 的值应设置为样本中捕获的RTP包的V、P、X、CC、M、PT和序列号字段。

字段bframe\_flag和repeat\_flag被保留在接收hint track中，并且必须为零。

extra\_flag和extra\_information\_length的语义与为RTP服务器hint track指定的语义相同。

指定了以下TLV框：rtphdrextTLV, rtpoffsetTLV, receivedCSRC。

如果设置了X\_bit，则需要一个rtphdrextTLV盒子来存储接收到的RTP报头扩展。

aligned(8) class rtphdrextTLV extends Box(‘rtpx’) {

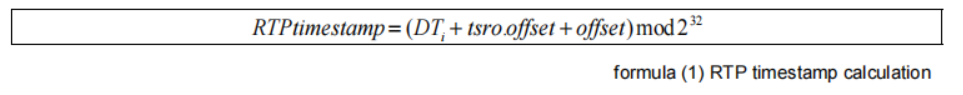
unsigned int(8) data[];

}

数据是特定于应用程序的原始RTP报头扩展。

rtpoffsetTLV box的语法在9.1.3.1中指定。

offset表示接收RTP包的RTP时间戳的32位有符号整数偏移量。设i为样本的样本数，DT(i)等于在8.6.1.2中指定的样本号i，tsro.offset。偏移量为参考接收hint样本项“tsro” box中的偏移量值，%为模操作。偏移值应符合以下公式：



注1:当每个接收hint sample代表所有接收到的具有相同RTP时间戳的RTP包时，可以将decoding time to sample box中的sample\_delta值设置为与RTP时间戳匹配。 换句话说，如上所述，可以将DT(i)设置为(the RTP timestamp –

tsro.offset – offset)(假设结果值大于或等于0)。这是推荐的。

注2:在所有RTP流中，RTP时间戳并不一定会随着RTP序列号的增加而增加，也就是说，数据包的传输顺序和播放顺序可能不相同。 例如，许多视频编码方案允许以回放顺序对之前和之后的图片进行双向预测。 由于样本以其解码顺序出现在track中，即，在RTP接收hint track的情况下，在接收顺序中，rtpoffsetTLV框中的偏移可以用来扭曲RTP时间戳，使其偏离采样时间DT(i)。

为了在Edit List Boxes中进行编辑，收到的RTP包的合成时间被推断为采样时间DT(i)和偏移量(如上所述)的和。

如果CSRC\_count的值不等于0，则可能会出现一个receivedCSRC box，用于存储每个RTP包收到的CSRC报头字段。 receivedCSRC box用四个字符代码' rcsr '标识

aligned(8) class receivedCSRC extends Box('rcsr') {

unsigned int(32) CSRC[]; //to end of the box

}

CSRC[]的表项数等于收到的SRTP报文的CC值。 CSRC[]的第n项等于RTP报头的第n个CSRC值。

#### 9.4.1.5 SDP information

movie和track的SDP信息都可以显示，如9.1.4所述。

### 9.4.2 RTCP Reception Hint Track

#### 9.4.2.1 Introduction

此子句指定了实时控制协议(RTCP)的接收hint track格式，该格式在IETF RFC 3550中定义。

RTCP用于在Internet协议上实时传输RTP会话的控制信息。 在streaming过程中，每个RTP流通常都有一个附带的RTCP流，它携带RTP流的控制信息。 一个RTCP接收提示跟踪携带一个RTCP流，并通过track引用与相应的RTP接收hint track相关联。

RTCP接收hint track的格式允许在hint示例中存储RTCP发送者报告。

RTCP发送者报告对于流记录特别有意义，因为它们反映了服务器的当前状态，例如，媒体计时(音频/视频包的RTP时间戳)与服务器时间(NTP格式的绝对时间)的关系。 了解这种关系对于回放记录的RTP接收hint track也是必要的，以便能够检测和纠正时钟漂移和抖动。

9.4.1.2中指定的时间戳同步框可以在播放文件之前纠正时钟漂移和抖动，因此，当timestamp\_sync等于2时，记录RTCP流是可选的。

对于RTCP接收hint track，没有等效的服务器hint track，因为RTCP消息是在传输过程中实时生成的。

#### 9.4.2.2 General

对于每个RTP接收hint track，应该有0个或一个RTCP接收hint track。 一个RTCP接收hint track应该包含一个track参考框，其中包括一个类型为“cdsc”的参考到相关的RTP接收hint track。

当i为一个样本的样本号时，8.6.1.2中所述的样本时间DT(i)为该数据包的接收时间。 接收时间的时钟源应与相关的RTP接收hint track相同。 RTCP接收hint track的Media Header Box中的时间尺度值应与关联的RTP接收hint track的Media Header Box中的时间尺度值相等。

#### 9.4.2.3 Sample Description Format

RTCP接收hint track的示例描述中的入口格式是' RTCP '。 除此之外，它在结构上与RTP的示例条目格式相同。 没有为附加数据字段定义的box。

#### 9.4.2.4 Sample Format

##### 9.4.2.4.1 Introduction

接收hint track中的每个示例表示一个或多个接收到的RTCP包。 每个示例包含两个区域:原始RTCP数据包和所需的任何额外数据。 请注意，样本的大小可以从样本大小表中得知，RTCP包的大小在包本身中表示(如RFC 3550中所述)，作为比包中32位单词的数量少一的计数。

##### 9.4.2.4.2 Syntax

aligned(8) class receivedRTCPpacket {

unsigned int(8) data[];

}

aligned(8) class receivedRTCPsample {

unsigned int(16) packetcount;

unsigned int(16) reserved;

receivedRTCPpacket packets[packetcount];

}

##### 9.4.2.4.3 Semantics

data[]包含一个原始的RTCP包，包括RTCP报告头，20字节的发送方信息块和任何数量的报告块。 请注意，每个RTCP数据包的大小是通过解析RTCP报头的16位长度字段来知道的。

paccketcount表示样本中包含的接收到的RTCP包的数量。

packets包含已接收到的RTCP数据包。

### 9.4.3 SRTP Reception Hint Track

#### 9.4.3.1 Introduction

本小节指定了IETF RFC 3711中定义的安全实时传输协议(SRTP)的接收hint track格式。

SRTP是基于Internet协议的实时媒体传输(RTP)的安全扩展。 每个SRTP流携带一种媒体类型，一个SRTP接收hint track携带一个SRTP流。 因此，一个视听节目的录音结果至少两个SRTP接收hint track。

SRTP接收hint track格式的设计遵循了RTP接收hint track的设计，重用了RTP接收hint track提供的大部分框架。 RTP和SRTP接收提示跟踪之间的主要区别是，对于SRTP接收hint track，实际的媒体负载是以加密的形式存储的，而对于RTP接收hint track，实际的媒体负载是不加密的。 SRTP接收hint tracck提供了额外的box来存储在回放时解密加密内容所需的信息。 此外，SRTP包报头的所有报头字段应与有效载荷一起存储，因为这些信息对于检查接收数据的完整性是必要的。 SRTP接收hint track通常与SRTCP接收hint tracck一起使用。

例如，SRTP接收提示轨道可用于存储受保护的移动电视内容。

#### 9.4.3.2 Sample Description Format

##### 9.4.3.2.1 Sample Description Entry

SRTP接收hint track的样例描述格式与RTP接收hint track的样例描述格式相同，只是样例条目名称从' rrtp '更改为' rsrp '，并且它可能包含额外的box:

class ReceivedSrtpHintSampleEntry() extends SampleEntry (‘rsrp‘) {

uint(16) hinttrackversion = 1;

uint(16) highestcompatibleversion = 1;

uint(32) maxpacketsize;

box additionaldata[];

}

字段和boxes与ReceivedRtpHintSampleEntry (' rrtp ')相同。 SRTP接收hint Track的每个样本描述条目的addtionaldata[]应包含一个ReceivedSsrc Box (' rssr ')。

另外，addtionaldata[]可能包含Received Cryptographic Context ID box和下面定义的Rollover Counter box。 此外，SRTP Process Box也应包括在additionaldata box中。 当内容被加密存储时，SRTP Process box中的完整性和加密算法字段指定应用到接收流的算法。 一个包含四个空格的条目($20$20$20$20)可以用来表示该算法是通过超出本文范围的方式定义的。

##### 9.4.3.2.2 Received Cryptographic Context ID Box

0或1个ReceivedCryptoContextIdBox，用4个字符的代码' ccid '标识，可能出现在SRTP接收hint track 的示例描述符条目的额外数据中。 恢复接收到的SRTP流的密码上下文的信息可能会存储在这里。

aligned(8) class ReceivedCryptoContextIdBox extends Box (‘ccid’) {

unsigned int(16) destPort;

unsigned int(8) ip\_version;

switch (ip\_version) {

case 4: // IPv4

unsigned int(32) destIP;

break;

case 6: // IPv6

unsigned int(64) destIP;

break;

}

}

destPort和destIP参数分别包含SRTP会话的端口号和IP地址(在接收到的IPv4或IPv6包中)，记录的SRTP报文是通过该SRTP会话接收的。 ip\_version包含4或6，分别代表IPv4和IPv6。

##### 9.4.3.2.3 Rollover Counter Box

0或1个RolloverCounterBox(用4个字符的代码' sroc '标识)可能出现在SRTP接收hint track的示例描述符条目的额外数据中。 通常，每次65536个SRTP包都会改变翻转计数器的值。

aligned(8) class RolloverCounterBox extends Box (‘sroc’) {

unsigned int(32) rollover\_counter;

}

rollover\_counter是一个非零整数，它给出了接收到的所有相关SRTP包的ROC字段的值。

注意:rollover counter(ROC)是SRTP流的加密上下文的一个元素，它取决于RTP流中包的绝对位置。 为了解密收到的SRTP报文，需要知道ROC值。 使用rollover counter box是可选的，因为RFC 4771将其定义为一种可选机制，以在SRTP包的身份验证标记中显式地显示ROC值。

#### 9.4.3.3 Sample and Packet Entry Format

SRTP接收hint track的样例格式和包入口格式都与9.4.1.3和9.4.1.4中定义的RTP接收hint track相同。 包有效载荷以接收到的方式存储在SRTP包中，即SRTP包中接收到的所有信息，不包括报头，也就是说，加密的有效载荷以及密钥标识符(MKI)和身份验证标记。

如果收到的SRTP报文的CSRC\_count的值不等于0，则与此receivedsrtpacket对应的extra\_data\_tlv应该包含一个receivedCSRC box (' rcsr ')。

### 9.4.4 SRTCP Reception Hint Tracks

#### 9.4.4.1 Introduction

本小节指定了IETF RFC 3711中定义的安全实时控制协议(SRTCP)的接收hint track格式。

SRTCP用于在Internet协议上实时传输SRTP会话的控制信息。 SRTCP对于SRTP的角色与RTCP对于RTP的角色相同，cf.， 9.4.2。 在流化过程中，每个SRTP流通常都有一个附带的SRTCP流，它携带SRTP流的控制信息。 一个SRTCP接收hint track携带一个SRTCP流，并通过track引用与相应的SRTP接收hint track相关联。

本小节指定了IETF RFC 3711中定义的安全实时控制协议(SRTCP)的接收hint track格式。

SRTCP接收hint track的格式允许在提示示例中存储SRTCP包，例如，SRTCP发送者报告。

SRTCP发送者报告对于流记录特别有意义，因为它们反映了服务器的当前状态，例如，媒体计时(音频/视频包的SRTP时间戳)与服务器时间(NTP格式的绝对时间)的关系。 了解这种关系对于回放记录的SRTP接收hint track也是必要的，以便能够检测和纠正时钟漂移和抖动。

9.4.1.2中指定的timestamp synchrony box可以在播放文件之前纠正时钟漂移和抖动，因此记录SRTCP流是可选的。

对于SRCTP接收hint track，没有等效的服务器hint track，因为SRTCP消息是在传输过程中动态生成的。

#### 9.4.4.2 General

对于每个SRTP接收hint track，应该有0个或一个SRTCP接收hint track。 一个SRTCP接收hint track应该包含一个track reference box，其中包括对相关SRTP接收hint track的类型为“cdsc”的引用。

当i为采样号a sample时，8.6.1.2中指定的采样时间DT(i)表示数据包的接收时间。 接收时间的时钟源应与相关的SRTP接收hint track相同。 SRTCP接收hint track的Media Header Box中的时间尺度值与关联的SRTP接收hint track的Media Header Box中的时间尺度值相等。

#### 9.4.4.3 Sample Description Format

SRTCP接收hint track的示例描述中的入口格式是'stcp'。 除此之外，它在结构上与RTCP的示例条目格式相同。 SRTCP hint track的加密和认证方法由对应SRTP hint track的SRTP Process box中的相应条目定义。

注意:为SRTP定义的等效ROC box对SRTCP来说是不必要的，因为SRTCP包包含一个显式的初始化向量。

#### 9.4.4.4 Sample Format

示例格式是9.4.2.4中定义的RTCP接收hint track的示例格式。

### 9.4.5 Protected RTP Reception Hint Track

#### 9.4.5.1 Introduction

这个规范定义了一种将媒体流标记为受保护的机制。 这是通过更改SampleEntry的4个字符代码，并添加包含保护机制细节和原始4个字符代码的box来实现的。 然而，在这种情况下，track是不受保护的; 它是一个包含受保护数据的“已清除” hint track。 本条款描述了接收hint track应该如何标记为携带受保护数据，使用类似的机制，并使用相同的box。

#### 9.4.5.2 Syntax

Class ProtectedRtpReceptionHintSampleEntry

extends RtpReceptionHintSampleEntry (‘prtp‘) {

ProtectionSchemeInfoBox SchemeInformation;

}

9.4.5.3 Semantics

方案信息(' sinf ') box应包含所应用的保护方案的详细信息。 这应该包括OriginalFormatBox，它应该包含四个字符代码' rrtp '(原始RTPReceptionHintSampleEntry框的四个字符代码)。

### 9.4.6 Recording Procedure

See Annex H.

### 9.4.7 Parsing Procedure

See Annex H.

# 10 Sample Groups

## 10.1 Random Access Recovery Points

#### 10.1.1.1 Definition

在一些编码系统中，可以随机进入一个流，并在解码了一些样本后实现正确的解码。 这就是所谓的渐进式解码刷新。 例如，在视频中，编码器可能会在流中编码内编码的宏块，这样它就知道在某一段时间内，整个图像由只依赖于该时间内提供的内编码宏块的像素组成。

这种允许渐进刷新的样本被标记为这个组的成员。 组的定义允许标记出现在周期的开始或结束。 但是，当与特定的介质类型一起使用时，这一组的使用可能被限制为只标记一端(即仅限制为正或负滚动值)。 卷组定义为具有相同卷距的一组样品。

#### 10.1.1.2 Syntax

class VisualRollRecoveryEntry() extends VisualSampleGroupEntry (’roll’)

{

signed int(16) roll\_distance;

}

class AudioRollRecoveryEntry() extends AudioSampleGroupEntry (’roll’)

{

signed int(16) roll\_distance;

}

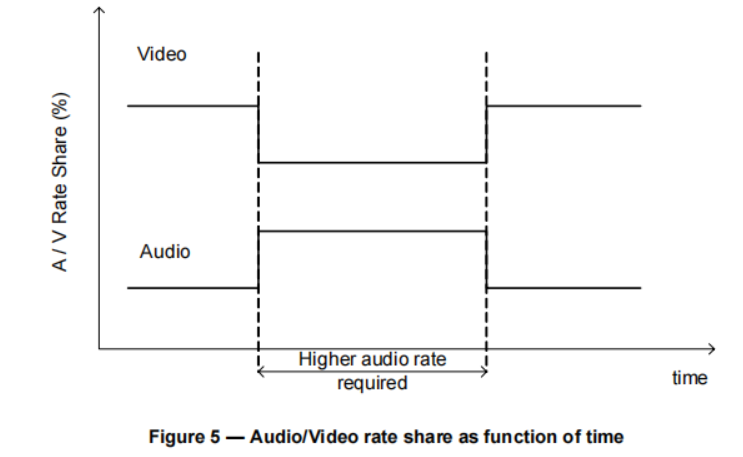
#### 10.1.1.3 Semantics

Roll\_distance是一个带符号的整数，它给出了为了正确解码一个样本必须解码的样本数量。 一个正的值表示在组成员的样本之后的样本数量，该样本必须被解码，以便在最后一次恢复时完成，即最后一个样本是正确的。 负值表示在采样前的采样数，该采样必须被解码，以便在标记的采样处完成恢复。 不能使用值0; 同步样例表记录了不需要恢复卷的随机访问点。

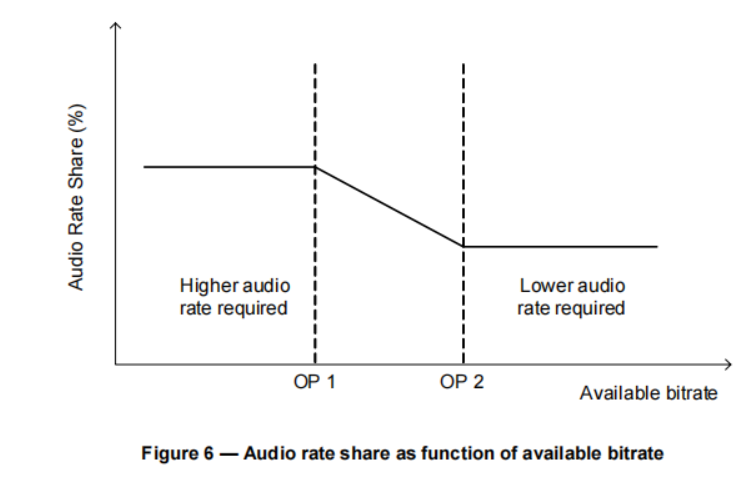
## 10.2 Rate Share Groups

### 10.2.1 Introduction

当多个流共享一个共同的带宽资源时，播放机和流媒体服务器使用速率共享指令来帮助动态分配比特率。 指令以样本组条目的形式存储在文件中，当不同比特率的可伸缩或可选媒体流与其他可伸缩或可选track相结合时适用。 指令是时间相关的，因为一个track中的样本可能与不同的样本组条目相关联。 在最简单的情况下，每个媒体和时间范围只指定一个目标速率共享值，如图5所示。



为了适应随可用比特率变化的速率共享值，可以指定多个操作范围。 例如，有人可能指出，在低可用比特率下，音频需要更高的百分比(比视频)。 从技术上讲，这是通过指定两个操作点来实现的，如图6所示。



操作点是根据总的可用带宽定义的。 对于更复杂的情况，可以指定更多的操作点。

除了目标速率共享值外，还可以指定特定媒体的最大和最小比特率，以及丢弃优先级。

### 10.2.2 Rate Share Sample Group Entry

#### 10.2.2.1 Definition

一个track的每个样本可以关联到(零或)多个样本组描述中的一个，每个样本组描述定义了一个速率共享信息的记录。 通常，相同的速率共享信息适用于许多连续的样本，因此定义两个或三个可在不同时间间隔使用的样本组描述就足够了。

分组类型“rash”(费率份额的缩写)被定义为费率份额信息的分组标准。 一个track的sample table box('stbl')中可以包含0个或一个分组类型'rash'的sample-to-group box('sbgp')。 如果使用hint track，则该track应位于hint track中，否则位于媒体track中。

目标速率共享可以为几个操作点指定，这些操作点是根据总可用比特率定义的，即应该共享的比特率。 如果只定义了一个操作点，则目标速率份额适用于所有可用的比特率。 如果定义了几个操作点，则每个操作点指定一个目标费率共享。 为第一个和最后一个操作点指定的目标速率共享值也分别指定可用比特率较低和较高的目标速率共享值。 两个操作点之间的目标利率份额被指定在这些操作点的目标利率份额之间的范围内。 一种可能方案是用线性插值估计。

#### 10.2.2.2 Syntax

class RateShareEntry() extends SampleGroupDescriptionEntry('rash') {

unsigned int(16) operation\_point\_count;

if (operation\_point\_count == 1) {

unsigned int(16) target\_rate\_share;

}

else {

for (i=0; i < operation\_point\_count; i++) {

unsigned int(32) available\_bitrate;

unsigned int(16) target\_rate\_share;

}

}

unsigned int(32) maximum\_bitrate;

unsigned int(32) minimum\_bitrate;

unsigned int(8) discard\_priority;

}

#### 10.2.2.3 Semantics

Operation\_point\_count是一个给出操作点数目的非零整数。

Available\_bitrate是一个正整数，定义了一个操作点(单位为千比特每秒)。 它是可以分配在共享中track的总可用比特率。 每一个entry应大于前一个entry。

Target\_rate\_share为整数。 非零值表示应该为每个操作点分配给媒体的可用带宽的百分比。 第一个(最后一个)操作点的值适用于比操作点本身更低(更高)的可用比特率。 各操作点之间的目标利率份额以相应操作点的目标利率份额为界。 零值表示没有提供有关首选费率份额百分比的信息。

“Maximum\_bitrate”为整数。 非零值表示(以千比特每秒为单位)应该为媒体分配带宽的上限阈值。 只有当会话中的所有其他媒体分别完成了它们的目标速率共享配额和最大比特率配额时，才应该分配高于最大比特率的比特率。 零值表示没有提供关于最大比特率的信息。

“Minimum\_bitrate”为整数。 非零值表示(以千比特每秒为单位)应该为媒体分配带宽的较低阈值。 如果分配的带宽对应较小的值，则不应分配比特率。 相反，应该优先考虑会话中的其他媒体或同一媒体的替代编码。 零最小比特率表示不提供有关最小比特率的信息。

Discard\_priority为整数，表示为满足目标速率共享、最大比特率和最小比特率设置的约束条件而丢弃track时，track的优先级。 Track按照丢弃优先级的顺序被丢弃，丢弃优先级最高的tracck优先被丢弃。

### 10.2.3 Relationship between tracks

定义速率共享信息的目的是帮助服务器或播放者从与其他track结合的track中提取数据。 请注意，如果服务器/播放器属于不同的可选组，它们可以同时流/播放曲目，并且可以在可选组中属于同一交换组的track之间切换。 默认情况下，如果没有定义替代组，所有曲目都是同时提供/播放的。

应该为每个track提供费率共享信息。 不包含速率共享信息的track有一个操作点，可以被视为丢弃优先级为128的恒定比特率track。 在这种情况下，目标速率共享、最小和最大比特率不适用。

相互交替的track应(在每一时刻)在相同的总可用比特率集上定义相同数量的操作点，并具有相同的丢弃优先级。 注意，操作点的数量和定义可能取决于时间。 交替track可能有不同的目标率份额，最小和最大比特率。

### 10.2.4 Bitrate allocation

速率共享信息的最大比特率，最小比特率，和目标速率共享可以组合为一个track。 在这种情况下，在考虑最大和最小比特率的影响之前，应先应用目标率份额来找到一个已分配的比特率。

当将带宽分配给几个轨道时，需要考虑以下几点:

1. 在这种情况下，所有track都有明确的目标费率共享值，它们的总和不等于100%，将它们视为权重，也就是说，将它们正常化。
2. 总分配不得超过总可用比特率。
3. 在可选track之间的选择中，所选的track应该是使可选组有一个最符合其目标速率份额的分配的track，或者是希望在不丢弃其他track的情况下分配最高比特率的tracck(见下文)。
4. track必须在其最小和最大比特率之间进行分配，否则将被丢弃。
5. track的分配应该与它们的目标费率份额一致，但这可能会被扭曲，以允许某些track达到其最小值，或在某些曲目达到其最大值的情况下。
6. 如果不能从每个可选组中分配一个track，那么应该按照优先级顺序丢弃track。
7. 每当活动track的操作集(从备用组中选择的操作集)发生变化或可用比特率发生变化时，必须重新计算分配。

## 10.3 Alternative Startup Sequences

### 10.3.1 Definition

可选的启动序列包含从同步样本或标记为“rap”样本分组的样本开始的特定时间段内的轨道样本子集，这些样本统称为下面的初始样本。 通过解码这个样本子集，可以比所有样本都被解码时更早地开始呈现样本。

一个“alst”的样本组描述条目表示在任何各自的替代启动序列中的样本数量，之后应该处理所有的样本。

Sample to Group Box的版本0或版本1可以与可选的启动序列样本分组一起使用。 如果使用了Sample to Group Box的版本1,grouping\_type\_parameter没有定义语义，但是对于grouping\_type\_parameter的特定值，应该使用相同的算法来派生可选启动序列。

使用可选启动序列的播放者可以执行如下操作。 首先，通过使用sync sample Box、sample\_is\_non\_sync\_sample标志(用于包含在曲目片段中的样本)或“rap”样本分组来标识要开始解码的初始同步样本。 然后，如果初始同步样本与类型为“alst”的样本组描述条目相关联，其中roll\_count大于0，玩家可以使用替代启动序列。 然后，玩家只解码那些映射到可选启动序列的样本，直到已解码的样本数量等于roll\_count。 之后，所有的样本都被解码。

### 10.3.2 Syntax

class AlternativeStartupEntry() extends VisualSampleGroupEntry (’alst’)

{

unsigned int(16) roll\_count;

unsigned int(16) first\_output\_sample;

for (i=1; i <= roll\_count; i++)

unsigned int(32) sample\_offset[i];

j=1;

do { // optional, until the end of the structure

unsigned int(16) num\_output\_samples[j];

unsigned int(16) num\_total\_samples[j];

j++;

}

}

### 10.3.3 Semantics

Roll\_count表示备选启动顺序中的样本数量。 如果roll\_count等于0，则相关的示例不属于任何可选的启动序列，并且first\_output\_sample的语义未指定。 每个可选启动序列映射到这个样例组条目的样例数量应该等于roll\_count。

First\_output\_sample表示备选启动序列中用于输出的第一个示例的索引。 启动备用启动序列的同步初始样本的索引是1，在解码顺序中，每个备用启动序列中的样本的索引都加1。

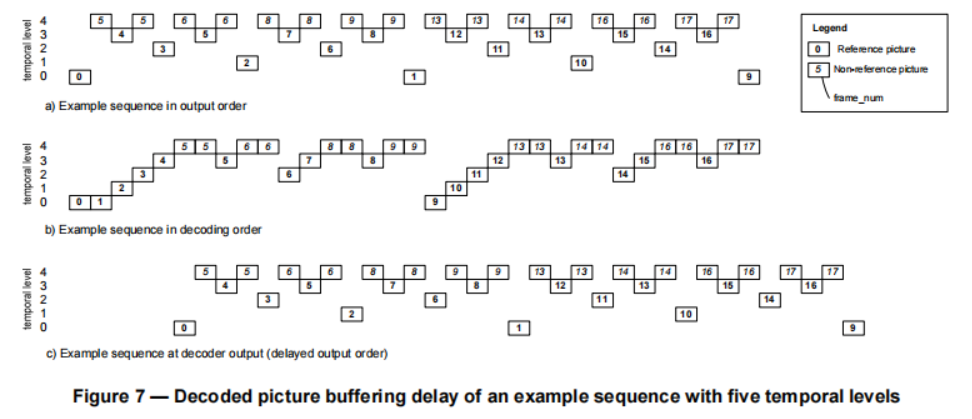
sample\_offset[i]表示备选启动序列中第i个样本的解码时间相对于从“Decoding Time to Sample Box”或“Track Fragment Header Box”中导出的样本的常规解码时间的delta。 启动备用启动序列的同步初始样例是它的第一个样例。

Num\_output\_samples [j]和num\_total\_samples[j]表示备选启动序列内的样本输出速率。 备选启动序列被分成k个连续的片段，每个片段都有一个恒定的样本输出速率，这个速率不等于相邻片段的采样输出速率。 第一部分从first\_output\_sample指示的示例开始。 Num\_output\_samples [j]表示备选启动序列第j个输出样本个数。 Num\_total\_samples [j]表示样本总数，包括不属于备选启动序列的样本， 从第j块的第一个样本输出到结束可选启动序列的样本的前一个样本(按组合顺序)，以及第(j+1)个样本的第一个输出样本的前一个样本。

### 10.3.4 Examples

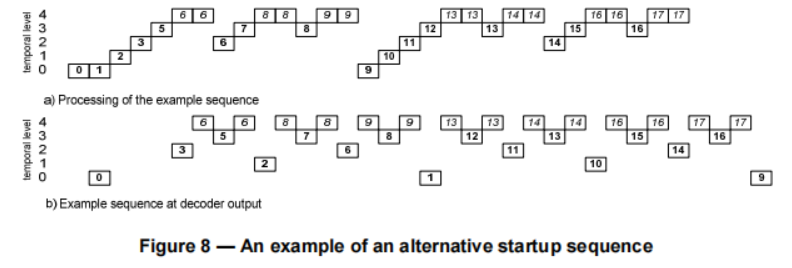
分层时序可伸缩性(例如，在AVC和SVC中)提高了压缩效率，但增加了解码延迟，因为解码后的图片从(de)编码顺序重新排序到输出顺序。 在一些研究中，深度时间层次结构在压缩效率方面已经被证明是有用的。 当时间层次较深且解码器的操作速度受到限制(不超过实际的时间处理速度)时，从解码开始到呈现开始的初始延迟很大，可能会对最终用户体验产生负面影响。

图7演示了一个典型的具有5个时间级别的分级可伸缩比特流。 图7a按输出顺序显示了示例序列。 框中的值表示图片的frame\_num值。 斜体部分为非参考图片，其余为参考图片。 图7b按解码顺序显示了示例序列。 图7c是假设输出时间轴与解码时间轴重合，且一幅图的解码时间间隔为一幅图时，按输出顺序排列的示例序列。 可以看出，流的回放开始的时间比流的解码开始的时间晚5个画面间隔。 如果以25 Hz采样图像，则图像间隔为40毫秒，回放延迟0.2秒。

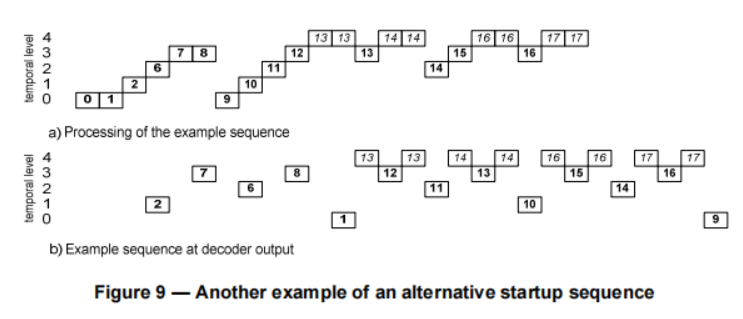


由于时间层次结构，只能解码序列开始部分图片的一个子集。 因此，渲染可以更快地开始，但在开始时显示的图像速率较低。 换句话说，播放者可以在初始启动延迟的持续时间和初始显示图像速率之间进行权衡。 图8和图9显示了两个可选启动序列示例，其中图7的位流子集被解码。

选择用于解码的样本和解码器输出分别如图8a和图8b所示。 frame\_num等于4的参考图片和frame\_num等于5的非参考图片不被解码。 在本例中，图片的呈现开始的时间间隔比图7早四个时间间隔。 当图片速率为25hz时，启动延迟保存时间为160msec。 在启动延迟中的保存伴随着比特流开始时较低的显示图片速率的缺点。



在图9的例子中，给出了另一种选择要解码的图片的方法。 省略了依赖于frame\_num = 3的图片的解码，也省略了第一组图片后半部分中非参考图片的解码。 从frame\_num等于2的示例中得到的解码图像是第一个输出的图像。 结果，第一组图片的输出图像速率是正常图像速率的一半，但显示过程开始的两帧间隔(25hz图像速率下的80毫秒)比图7所示的传统解决方案早。



## 10.4 Random Access Point (RAP) Sample Grouping

### 10.4.1 Definition

同步采样被指定为一个随机接入点，在此之后，按解码顺序的所有采样都可以被正确解码。 然而，编码一个“开放的”随机接入点是可能的，在此之后，所有按输出顺序的样本都可以被正确解码，但是一些按解码顺序在随机接入点之后和按输出顺序在随机接入点之前的样本不需要被正确解码。 例如，启动一组开放图片的内部图片可以在解码顺序后跟随(bi-)预测图片，但在输出顺序上在内部图片之前; 虽然如果解码从内部图像开始，它们可能不能被正确解码，但它们是不需要的。

这样的“open”随机访问样本可以被标记为这个组的成员。 由该组标记的样本必须是随机访问点，也可以是同步点(也就是说，由同步样本表标记的样本不需要被排除)。

### 10.4.2 Syntax

class VisualRandomAccessEntry() extends VisualSampleGroupEntry (’rap ’)

{

unsigned int(1) num\_leading\_samples\_known;

unsigned int(7) num\_leading\_samples;

}

### 10.4.3 Semantics

Num\_leading\_samples\_known = 1表示已知该组中每个样本的前导样本数，该数由num\_leading\_samples指定。 领先示例是与“open”随机访问点(RAP)相关的示例。 呈现顺序在RAP之前，解码顺序在RAP或其他领先样本之后，当从RAP开始解码时，该样本不能被正确解码。

Num\_leading\_samples指定该组中每个样本的前导样本的数量。 当num\_leading\_samples\_known等于0时，该字段应该被忽略。

## 10.5 Temporal level sample grouping

### 10.5.1 Definition

许多视频编解码器支持时间可伸缩性，可以提取一个或多个可以独立解码的帧子集。 一个简单的例子是为一个具有常规I帧间隔的比特流提取I帧，例如，IPPPIPPP…，其中每4张图片就是一个I帧。 对于更低的帧速率，也可以提取这些I帧的子集。 使用层次B或P框架可以构建具有多个时间层次的更复杂的情况。

时态级别样本分组(“tele”)提供了一种独立于编解码的样本分组，可用于根据时态级别对跟踪(和潜在的跟踪片段)中的样本(访问单元)进行分组，其中一个时态级别的样本与更高时态级别的样本没有编码依赖关系。 时间级别等于样本组描述索引(取值1、2、3等)。 仅包含从第一时间层到更高时间层的访问单元的比特流仍然符合编码标准。

根据时间级别分组便于提取时间子序列，例如使用8.16.4中的Subsegment Indexing box。

### 10.5.2 Syntax

class TemporalLevelEntry() extends VisualSampleGroupEntry('tele')

{

bit(1) level\_independently\_decodable;

bit(7) reserved=0;

}

### 10.5.3 Semantics

样本组中样本的时间水平等于样本组描述指数。 Level\_independently\_decodable是一个标志。 1表示此级别的所有样本与其他级别的样本之间没有编码依赖关系。 0表示不提供任何信息。

# 11 Extensibility

## 11.1 Objects

本规范中定义的标准对象由一个32位值标识，该值通常是来自ISO 8859-1字符集的四个可打印字符的集合。

为了允许格式的用户扩展、存储新的对象类型，以及允许按照该规范格式化的文件在特定的分布式计算环境中进行互操作，有一种类型映射和一种类型扩展机制可以组成一对。

在分布式计算中常用的是uuid(通用唯一标识符)，长度为16字节。 这里指定的任何规范类型都可以通过将4字节类型值与12字节ISO保留值0xXXXXXXXX-0011-0010-8000-00AA00389B71组合在一起，直接映射到UUID空间。 4个字符的代码替换前面数字的“XXXXXXXX”。 ISO将这些类型标识为本规范中使用的对象类型。

用户对象使用转义类型' uuid '。 它们在上述第6.2款中有记录。 在大小和类型字段之后，有一个完整的16字节UUID。

希望将每个对象都视为具有UUID的系统可以使用以下算法:

size := read\_uint32();

type := read\_uint32();

if (type==‘uuid’)

then

uuid := read\_uuid()

else

uuid := form\_uuid(type, ISO\_12\_bytes);

类似地，当将一组对象线性化为按本规范格式化的文件时，应用以下方法:

write\_uint32( object\_size(object) );

uuid := object\_uuid\_type(object);

if (is\_ISO\_uuid(uuid) )

write\_uint32( ISO\_type\_of(uuid) )

else {

write\_uint32(‘uuid’);

write\_uuid(uuid);

}

包含本规范中使用' uuid '转义和完整uuid编写的box的文件是不兼容的; 系统不需要识别使用' uuid '和ISO uuid编写的标准box。

## 11.2 Storage formats

包含元数据的主文件可以使用其他文件来包含媒体数据。 这些其他文件可能包含来自各种标准的头声明，包括这个标准。

如果这样的辅助文件中有一个元数据声明集，那么该元数据就不是整个表示的一部分。 这允许通过构建新的元数据并引用媒体数据(而不是复制它)将小的表示文件聚合成更大的整体表示。

对这些其他文件的引用不需要使用这些文件中的所有数据; 这样，可能会使用媒体数据的子集，或者忽略不需要的标题。

11.3 Derived File formats

该规范可以作为特定文件格式的基础，用于限制用途:例如，MPEG-4的MP4文件格式和Motion JPEG 2000文件格式都是从它派生出来的。 在编写派生规范时，必须指定以下内容:

新格式的名称，以及文件类型框的brand和兼容性类型。 通常会使用新的文件扩展名、新的MIME类型和Macintosh文件类型，尽管这些文件的定义和注册超出了本规范的范围。

使用的任何模板字段都必须显式声明; 它们的使用必须符合这里的规范。

必须定义样例描述中使用的确切的“编码名”和“协议”标识符。 还必须定义这些代码点标识的样例的格式。 然而，将新的编码系统适应于现有的框架(例如MPEG-4系统框架)可能比在这个级别上定义新的编码点更可取。 例如，一个新的音频格式可以使用一个新的编码名，或者可以使用' mp4a '并在MPEG-4音频框架中注册新的标识符。

可以定义新的box，尽管不鼓励这样做。

如果派生的规范需要一个新的音轨类型，而不是视觉的或音频的，那么必须注册一个新的处理程序类型。 必须标识此track所需的媒体标头。 如果它是一个新box，则必须定义它并注册它的box类型。 一般来说，大多数系统都可以使用现有的track类型。

任何新的轨迹引用类型都应该注册和定义。

如上面定义的，样例描述格式可以扩展为可选或必需的框。 执行此操作的通常语法是定义一个具有特定名称的新框，扩展(例如)Visual Sample Entry，并包含新box。

# Annex A

(informative)

**Overview and Introduction**

## A.1 Section Overview

本节介绍文件格式，有助于读者理解文件格式的整体概念。 这一节形成了本规范的一个内容丰富的附录。

## A.2 Core Concepts

在文件格式中，整个演示称为电影。 它逻辑上被划分成track; 每个track代表一个媒体的定时序列(例如，视频帧)。 在每个track中，每个定时单元被称为一个样本; 这可能是一帧视频或音频。 样本隐式地按顺序编号。 请注意，一帧音频可能会解压成一系列音频样本(在这个意义上，这个词在音频中使用); 通常，该规范使用“样本”一词来表示一个定时帧或数据单元。 每个track有一个或多个样本描述; track中的每个样本都是通过引用绑定到描述。 该描述定义了如何对样本进行解码(例如，它识别所使用的压缩算法)。

与其他许多多媒体文件格式不同，这种格式和它的祖先分离了几个经常链接的概念。 理解这种分离是理解文件格式的关键。 特别是:

文件的物理结构并不与媒体本身的物理结构绑定。 例如，许多文件格式“帧”媒体数据，将头文件或其他数据立即放在视频的每一帧之前或之后; 这个文件格式不这样做。

无论是文件的物理结构，还是媒体的布局，都与媒体的时间顺序无关。 视频帧不需要按时间顺序(尽管它们可能是)放置在文件中。

这意味着存在描述媒体位置和时间的文件结构; 这些文件结构允许(但不要求)按时间顺序排列的文件。

一致文件中的所有数据都封装在box中(在该文件格式的前身中称为原子)。 box结构之外没有数据。 所有元数据，包括定义媒体位置和时间的元数据，都包含在结构化的box中。 本规范定义了box。 媒体数据(例如视频帧)被这个元数据引用。 媒体数据可以在同一个文件中(包含在一个或多个box中)，也可以在其他文件中; 元数据允许通过url引用其他文件。 媒体数据在这些辅助文件中的位置完全由主文件中的元数据描述。 它们不需要按照这个规范进行格式化，尽管它们可能是; 例如，在这些二级媒体文件中可能没有box。

track可以是各种各样的。 这里有三个很重要。 视频track包含的样本是可视的; 音频track包含音频媒体。 hint track是相当不同的; 它们包含用于流媒体服务器的指令，说明如何从文件中的媒体track形成流协议的数据包。 当读取文件进行本地回放时，可以忽略提示轨道; 它们只与流媒体相关。

## A.3 Physical structure of the media

在示例表中可以找到定义媒体数据布局的box。 这包括数据引用、样本大小表、样本到数据块表和数据块偏移表。 在它们之间，这些表允许定位轨迹中的每个样本，并知道其大小。

数据引用允许在辅助媒体文件中定位媒体。 这允许从特定文件的媒体的库建立一个组成成份，而不是实际复制媒体到一个单一的文件。 例如，这极大地方便了编辑。

表被压缩以节省空间。 此外，预计交错将不是一个样本接着一个样本，而是一个track的几个样本将同时出现，接着是另一个track的一组样本，以此类推。 这些同一个track连续的样本集合称为块。 每个块都有一个到其包含文件的偏移量(从文件的开始)。 在块中，样本是连续存储的。 因此，如果一个块包含两个样本，可以通过将第一个样本的大小加到块的偏移量来找到第二个样本的位置。 块偏移表提供偏移量; 样本到块表提供了从样本号到块号的映射。

请注意，在块之间(但不是在它们内部)可能有“死亡空间”，不被媒体数据引用。 因此，在编辑过程中，如果某些媒体数据不需要，可以直接不引用; 删除数据时不需要复制数据。 同样地，如果媒体数据位于格式化为“外部”文件格式的辅助文件中，则可以简单地跳过由该外部格式强制的头文件或其他结构。

## A.4 Temporal structure of the media

文件中的计时可以通过一些结构来理解。 这部电影和每首歌曲都有一个时间表。 这定义了一个时间轴，该时间轴具有每秒的节拍数。 通过适当选择这个数字，可以实现精确的定时。 通常对于一个音频track，这是音频的采样率。 对于视频，应该选择合适的比例。 例如，30000的媒体TimeScale和1001的媒体采样持续时间准确定义了NTSC视频(通常，但不正确，称为29.97)，并以32位提供了19.9小时的时间。

track的时间结构可能会受到编辑列表的影响。 这提供了两个关键的功能:在整个movie中，track时间线部分的移动(和可能的重用)位置，以及插入“空白”时间，即所谓的空编辑。 特别要注意的是，如果一个track不是在presentation的开始部分开始，那么就需要一个初始的空编辑。

每个轨道的总持续时间在报头中定义; 这提供了一个有用的轨迹总结。 每个示例都有一个定义的持续时间。 一个样本的确切呈现时间(它的时间戳)是通过对前面的样本的持续时间求和来定义的。

## A.5 Interleave

文件的时间和物理结构可以对齐。 这意味着在使用时，媒体数据在其容器中按时间顺序有其物理顺序。 此外，如果多个声道的媒体数据包含在同一个文件中，那么这些媒体数据将被交叉使用。 通常，为了简化单道媒体数据的读取，并保持表的紧凑，这种交错是在一个合适的时间间隔(例如1秒)进行的，而不是一个样本接着一个样本。 这样可以减少块的数量，从而使块偏移表变小。

## A.6 Composition

如果多个audio track包含在同一个文件中，它们将被隐式地混合用于播放。 这种混合受整体track音量和左右平衡的影响。

同样地，视频轨道是由它们的层号(从后到前)和它们的合成模式组成的。 此外，每个track可以用一个矩阵变换，也可以用矩阵变换整个电影。 这既允许简单的操作(如像素加倍，90º旋转的校正)，也允许更复杂的操作(如剪切，任意旋转)。

派生规范可以使用更强大的系统(例如MPEG-4 BIFS)来超越音频和视频的这种默认组合。

## A.7 Random access

介绍搜索的操作步骤。 查找主要是通过使用sample table box中包含的子box来完成的。 如果有编辑列表，也必须咨询它。

如果你想寻找一个时间T的给定track，而T在movie header box的时间尺度内，你可以执行以下操作:

1. 如果track包含编辑列表，则通过遍历编辑确定哪个编辑包含时间T。然后，在movie时间尺度中编辑的开始时间必须从时间T中减去，以生成T'，即在电影时间尺度中编辑的持续时间。 然后将T’转换为track的媒体的时间尺度，生成T”。 最后，通过将编辑的媒体开始时间加到T”来计算要使用的媒体尺度中的时间。
2. track的time-to-sample box表示该track的采样时间与哪个采样时间相关联。 使用此box查找给定时间之前的第一个样本。
3. 位于步骤1中的示例可能不是同步示例。 同步样本表指出哪些样本实际上是随机访问点。 使用此表，您可以定位在指定时间之前的第一个同步示例。 同步样例表的缺失表明所有样例都是同步点，这使得这个问题很容易解决。 在查看了同步示例表之后，您可能希望找到与步骤1中找到的示例最接近的结果示例。
4. 在这一点上，你知道将用于随机访问的样本。 使用sample-to-chunk表确定该样本位于哪个数据块中。
5. 知道哪个块包含有问题的样本，使用块偏移量框来找出该块从哪里开始。
6. 从这个偏移量开始，您可以使用包含在sample-to-chunk box和sample size box中的信息来确定该样本在该数据块中的位置。 这就是我们想要的信息。

## A.8 Fragmented movie files

本节介绍一种可以在ISO文件中使用的技术，在这种技术中，在movie中构建单个Movie Box是非常繁琐的。 至少在以下情况下会出现这种情况:

* 录制。 此时，如果一个录制应用程序崩溃，磁盘耗尽，或其他一些事件发生，在它写了很多媒体到磁盘，但在它写Movie Box之前，录制的数据是不可用的。 这是因为文件格式要求将所有元数据(Movie Box)写入文件的一个连续区域。
* 录制。 在嵌入式设备上，特别是静止的相机上，没有RAM来缓冲movie box的可用存储空间大小，当电影关闭时重新计算它太慢了。 崩溃的风险也同样存在。
* HTTP快速启动。 如果movie的大小合适(就movie Box而言，如果不是time的话)，那么在快速开始之前，movie Box可能需要一段不舒服的下载时间。

Movie的基本“形状”是在初始的movie Box中设置的:track的数量、可用的示例描述、宽度、高度、组成等。 然而，“Movie Box”并不包含movie整个期间的信息; 特别是，它的track中可能有很少或没有样本。

在这个最小的或空的movie中，添加了额外的样本，这种结构称为movie片段。

基本的设计理念与Movie Box相同; 数据不是“framed”。 然而，设计是这样的，如果需要的话，它可以被视为一个“framing”设计。 这些结构可以很容易地映射到Movie Box，因此可以将碎片化的表示重写为单个Movie Box。

这种方法是为每个示例设置默认值，包括全局(每个track一次)和每个片段。 只有那些具有非默认值的片段才需要包含这些值。 这使得常见的情况-规则的、重复的、结构-紧凑，不禁用增量构建有变化的movie。

普通的Movie Box设置movie的结构。 它可能出现在文件中的任何地方，不过对于读取器来说，它最好出现在片段之前。 (这不是一条规则，因为对Movie Box进行微小的更改，迫使它写到文件的末尾是不可能的)。 这个movie的box:

* 必须以自己的方式表示一个有效的电影(尽管轨道可能根本没有样本);
* 有一个盒子，表明碎片应该被发现和使用;
* 用于包含完整的编辑列表(如果有的话)。

请注意，不能理解片段的软件将只播放这个最初的movie。 能够理解片段并获得非片段movie的软件不会扫描片段，因为片段指示box不会被找到。

# Annex B

**(informative)**

**Patent Statements**

国际标准化组织和国际电工委员会(IEC)提请注意，其声称遵守ISO/IEC 14496和ISO/IEC 15444的这一部分可能涉及专利的使用。

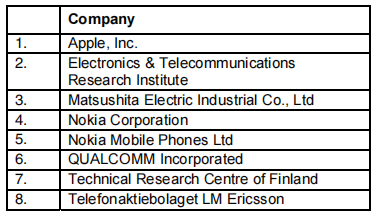
ISO和IEC对这些专利权的证据、有效性和范围不持任何立场。

这些专利权的持有者已经向ISO和IEC保证，他们愿意在合理和非歧视性的条款和条件下与世界各地的申请人谈判许可证。 在这方面，这些专利权持有人的声明已在ISO和IEC注册。 有关资料可向下列公司索取。

请注意，ISO/IEC 14496和ISO/IEC 15444本部分的某些要素可能是本附件中所确定的专利以外的专利主题。 ISO和IEC不负责识别任何或所有此类专利权

ISO (www.iso.org/patents)和IEC (http://patents.iec.ch)维护与其标准相关的在线专利数据库。 我们鼓励用户在数据库中查阅有关专利的最新信息。

请注意，适用于ISO基础媒体文件格式的专利声明可能不适用于ISO/IEC 15444-3 (Motion jpeg2000)的实施。 ISO/IEC 15444-3使用ISO/IEC 15444-12 (ISO Base Media File Format)的子集。



# Annex C

(informative)

**Guidelines on deriving from this specification**

## C.1 Introduction

本附件提供了详实的文本来解释如何从ISO基础媒体文件格式派生出特定的文件格式。

ISO/IEC 14496-12 | ISO/IEC 15444-12 ISO Base Media Format定义了文件格式的基本结构。 特定于媒体和用户定义的扩展可以在派生自ISO基础媒体文件格式的其他规范中提供。

C.2 General Principles

### C.2.1 General

许多现有的文件格式使用ISO基础媒体文件格式，尤其是MPEG-4的MP4文件格式(ISO/IEC 14496-14)和Motion JPEG 2000 MJ2文件格式(ISO/IEC 15444-3)。 当考虑从ISO基础媒体文件格式派生的新规范时，所有现有的规范都应该作为示例和定义和技术的来源。 与注册机构进行检查，以发现可能已经存在的内容以及存在哪些规范。

特别是，如果现有规范已经涵盖了特定媒体类型如何以文件格式存储(例如，MPEG-4视频在MP4中)，那么就应该使用该定义，而不应该发明新的定义。 通过这种方式，共享技术的规范也将共享该技术如何表示的定义。

对文件中存在的其他信息尽可能宽容; 指示无法识别的box和媒体可以被忽略(而不是“应该被忽略”)。 这允许创建混合文件，从多个规范中提取，并创建能够处理多个规范的多格式播放器。

在对该规范进行分层时，值得注意的是，对于较低的(第12部分)规范，有一些特征是有意为之的“参数”，需要加以说明。 同样，第12部分文件格式规范的一些特性是内部的，其他规范很少讨论这些特性。 当然，在两者之间的灰色地带也有一些特征。

派生规范最好完全按照第12部分文件格式的参数来编写; 样本是什么，它的时间戳意味着什么，等等。 在派生规范中提到特定的现有box通常会被证明是错误的，除非在有限的情况下(例如，添加user-data box或扩展box)。

### C.2.2 Base layer operations

应该可以在不了解任何潜在的派生规范的情况下对第12部分文件执行一些操作。 这些操作可能包括明显的读取track、为示例查找数据和计时、以及它们的示例描述和track类型，等等。 这可以通过文件格式检查器或参考软件之类的通用库来实现。

不太明显的是对文件的一类操作:

1. re-interleaving数据; 将媒体数据按时间顺序排列，将不同track的样本分组成合理大小的块，并将块交叉排列;
2. 通过将数据从外部文件复制到新文件，使使用数据引用的文件自包含;
3. 去除自由空间原子，压缩原子结构;
4. 从“mdat”原子中删除似乎没有被track或元数据原子引用的数据;
5. 删除没有关联样本的样本条目;
6. 删除没有相关样本的样本组;
7. 提取一些track并仅使用这些track制作一个新文件(例如，从音频/视频presentation中提取一个audio track);
8. 插入或删除影片片段，或重新分割影片。

当然，这个列表并不详尽。

## C.3 Boxes

您可以向文件格式添加box，但要注意它们与其他box的交互方式。 特别是，如果它们“交叉链接”到现有的box中，您可能无法将这些文件标记为符合第12部分。

你必须注册所有的新box，除了那些使用' uuid '类型的box。 同样，您应该注册编解码器(示例条目)名称、brand、跟踪引用类型、处理程序(媒体类型)、组类型和保护方案类型。 在没有注册的情况下使用这些标识符确实是一个坏主意，因为可能会发生冲突——或者其他人可能注册具有不同含义的相同标识符。

如果可以使用简单的四字符代码，你不应该使用' UUID转义'(保留的ISO UUID模式0xXXXXXXXX-0011-0010-8000-00AA00389B71，其中四字符代码代替了X)编写一个盒子，理想情况下，你不应该设计使用UUID盒子; 如果可能的话，最好将数据放在已知的文件格式的“扩展点”中，或者注册一个新的box类型，如果真的需要的话。

不要忘记ISO文件中的所有数据必须是，或包含在，box中。 你可以引入一个签名，但它必须“看起来”像一个box。

如果可能的话，不要要求您定义的任何现有或新box处于特定的位置。 例如，现有的jpeg2000规范要求一个签名box，并且签名box必须在文件的第一个位置。 如果另一个规范也定义了一个签名box，并且要求它是第一个，那么就不能构造符合两个规范的文件。

它必须能够通过查找box长度来“遍历”文件的顶层。 不要忘记“隐含长度”在文件级别是允许的。

除非绝对不可避免，数据框应该包含其中一个数据(例如字段中)，或者其他的数据box，但不能同时包含两个数据box。 所有包含数据的box都应该是完整的box，以便以后对语法和语义进行更改。 包含其他box的box被称为容器box，通常是普通的(非完整的)box，因为如果它们被记录为只包含box，它们的语义将永远不会改变。

## C.4 Brand Identifiers

### C.4.1 Introduction

本节介绍在file-type box中brand标识符的使用，包括:

* 引入一个新brand。
* 播放者的行为取决于brand。
* 在创建ISO Base Media文件时设置brand。

brand识别一个规范，并做出一组简单的声明:

1. 该文件符合所识别的规范的所有要求;
2. 该文件不包含与已识别的规范相反的内容;
3. 实现该规范的读取器可能读取、解释并可能呈现该文件，忽略它无法识别的数据。

因此，规范应该说明(如果他们需要一个brand)“标识符合该规范的文件的brand是XXXX”，并注册该brand。

### C.4.2 Usage of the Brand

为了识别文件遵从的规范，brand被用作文件格式中的标识符。 这些brand都赢得了File Type Box。

例如，一个brand可能表示:

1. 可能出现在文件中的编解码器,
2. 每个编解码器的数据是如何存储的，
3. 应用于文件的约束和扩展名。

如果有必要制定不完全符合现有标准的新规范，则可以注册新brand。 例如，3GPP允许在文件格式中使用AMR和H.263。 由于这些编解码器在当时的任何标准中都不受支持，3GPP规定了在ISO基础媒体格式中使用SampleEntry和模板字段，并定义了这些编解码器引用的新box。 考虑到文件格式在未来将被广泛使用，预计将需要更多的brand。

brand不是添加剂; 他们独立。 你不能说:“这个brand表明还(also)需要对Y的支持”，因为“also”没有所指。

重写文件的系统应该删除它们不认识的brand，因为它们不知道该文件是否仍然符合该brand的要求(例如，重新交错(interleaving)一个文件可能会使它不符合要求某种交错样式的规范)。

请注意，主brand通常意味着文件扩展名，而文件扩展名又意味着MIME类型。 但这些都不是规则。 此外，当在MIME类型下服务时，不要忘记MIME类型可以接受参数，并且兼容brand的列表通常对接收系统很有用。

### C.4.3 Introduction of a new brand

一个新的brand可以定义，如果符合新的规范必须指出。 这通常意味着一个新brand的定义至少需要满足以下条件之一:

1. 使用任何现有brand都不支持的编解码器。
2. 在组合中使用任何现有brand不支持的多个编解码器。 此外，只有当播放器支持对文件中的所有媒体进行解码时，才允许播放文件。
3. 使用用户特定的约束和/或扩展(框，模板字段等)。

但是，该文件格式同时包含一个major\_brand字段和一个compatible\_brands数组。 这些字段由文件作者和第12部分规范拥有。 不要编写关于这些字段的规范，而仅仅是关于brand和它们的含义。 特别是，不要声明major\_brand字段(“符合此规范的文件必须将major\_brand设置为XXXX”)，因为一个文件永远不可能符合以这种方式编写的两个这样的规范，而且还会阻止别人从您的规范派生一个规范。 但是，可以定义仅允许作为兼容brand的brand。

然而，brand可以作为一种追踪剂。 拥有一个没有任何要求的brand是完全合法的，它被放在一个文件中作为“我曾在那里”的点(或者严格地说，“这个brand要求该文件最后由ZZZZ编写”)。

### C.4.4 Player Guideline

如果compatible\_brands列表中存在多个brand，并且播放器支持一个或多个brand，那么播放器应该播放符合这些规范的文件的那些方面。 在这种情况下，播放器可能无法解码不支持的媒体。

### C.4.5 Authoring Guideline

如果作者想要创建一个遵从多个规范的文件，需要考虑以下事项:

1. 文件中必须没有任何与brand标识的规范相反的内容。 例如，如果一个规范要求文件是自包含的，那么该规范的brand标识不能用于非自包含的文件。
2. 如果作者认为只符合其中一种规范的播放器只能播放符合该规范的媒体，那么就可以指出该brand。
3. 如果作者要求播放来自多个规范的媒体，那么就需要一个新brand，因为这代表了对播放器的新一致性要求。

### C.4.6 Example

在本节中，我们以一个可以定义新brand的例子为例。

首先，我们解释一下现有的两个brand。 如果brand' 3gp5 '在compatible\_brands的列表中，它表明该文件包含3GPP TS 26.234 (Release 5)中定义的媒体，按照标准指定的方式。 例如，' 3gp5 'brand的文件可能包含H.263。 同样，如果brand' mp42 '在compatible\_brands的列表中，则表明该文件包含ISO/IEC 14496-14中以指定的方式定义的媒体。 例如，“mp42” brand的文件可能包含MP3。 然而，“3gp5” brand不支持MP3。

假定该文件包含H.263和MP3，并将' 3gp5 '和' mp42 '作为compatible\_brands。 如果播放器只遵从“3gp5”而不支持MP3，建议播放器的行为是只播放H.263。 如果内容的作者不希望出现这种行为，则定义一个新brand，以表明文件中同时支持H.263和MP3。 通过在compatible\_brands列表中指定新定义的brand，可以防止上述行为，并且该文件只能在播放器同时支持H.263和MP3的情况下播放。

## C.5 Storage of new media types

在如何存储新媒体类型的定义中有两种选择。

首先，如果希望或接受MPEG-4系统结构，那么:

1. 一个新的ObjectTypeIndication应该被请求和使用;
2. 这个编解码器的解码器特定信息应该定义为MPEG-4描述符;
3. 应该为该媒体定义访问单元格式。

然后媒体使用文件格式中的MPEG-4代码点; 例如，一个新的视频编解码器将使用一个类型为“mp4v”的样本条目。

如果MPEG-4系统层不适合或不希望使用，则:

1. 应请求并使用新的四字符样本输入代码;
2. 解码器需要的任何附加信息都应该定义为存储在样本条目中的box;
3. 应该为该媒体定义文件格式示例格式。

注意，在第二种情况下，注册机构还将分配一个objecttypeindication用于MPEG-4系统。

## C.6 Use of Template fields

模板字段在文件格式中定义。 如果派生规范中使用了，则使用必须与基定义兼容，并且基定义的使用必须明确记录。

## C.7 Tracks

### C.7.1 Data Location

一个track是一个采样的时间序列; 每个示例由其数据(包含的字节)、长度和位置定义。 样本的长度和数据是文件格式的外部参数; 字节的位置不是。

存储数据的确切方式是第12部分文件格式的内部内容。 在定义你的格式中的样本是什么时，你应该定义样本的长度和数据。

但是，你不应该提到下面的box，因为它们的结构是开放的，它们存储的信息可能会以其他方式存储(例如，样本大小信息可能在stsz box，stz2 box，或一个movie片段):

sample size (stsz), compact sample size(stz2)

事实上，样本存储在一个track的连续运行的样本中; 这些运行被称为块，是交叉的来自不同track的块。 但是文件可能会被重新交错或重新分组; 下面的box是关于如何进行分组的:

chunk offsets (stco or co64), sample-to-chunk (stsc)

最关键的是，在第12部分文件中定位数据必须通过这些box(或在电影片段中的等效box)完成。 media data box(' mdat ')仅仅是一个可能的位置，并且就其本身而言，它只能被认为是一个由无法识别的位元组成的无序包。 不能保证media data box中所需要的材料是box中唯一的数据，或者是按照任何特定的顺序排列的数据，特别是在使用数据引用时，甚至根本不能保证media data box中有任何特定的样本。 在派生规范中提到media-data (' mdat ')box几乎肯定是错误的，试图定义(或假设)它的结构篡夺了第12部分规范，是错误的。

在集成规范中，完全可以要求某种样式、持续时间或大小的交错(“该规范要求文件是自包含的，并且媒体数据按解码时间顺序排列，交错的粒度不超过一秒”)。

### C.7.2 Time

类似地，在文件格式中，通过解码时间戳和可选的组合时间戳对样本进行时间参数化。 你应该定义这些对你的媒体意味着什么。 但是，存储这些文件的方式仍然是第12部分文件格式的内部内容。

但是，你不应该提及下列box，因为它们的结构是可以更改的，而且它们存储的信息可能会以其他方式存储:

time-to-sample box (stts), composition offsets (ctts)

同样，编辑的时间结构效果应该由文件格式保存，但是第12部分文件简化器可以，例如，合并两个相邻的编辑，实际上属于一起(例如，两个空编辑，或一个编辑选择时间A - B后，一个编辑选择时间B-C)。

### C.7.3 Media Types

在第12部分规范中有许多媒体类型:视频、音频、元数据等等。 它们由跟踪处理程序类型和特定于媒体的媒体标头表示。 可以注册新的媒体处理程序，但很少需要这样做。 它可能是需要的，例如，如果需要一个轨道类型，例如，实验室仪器跟踪，或“timed aroma(香气)”跟踪。 对登记机关也应当进行检查; 所需的处理程序可能已经在另一个派生规范中定义。

### C.7.4 Coding Types

示例条目的名称标识所使用的编码格式。 这是第12部分规范参数化的主要方式之一; AVC (MPEG-4第10部分)以' avc1 '为例，作为示例条目类型。 为编解码器定义这个名称并注册它，然后在这个编解码器的示例条目中定义什么是额外的box，这是第12部分格式使用的主要方法。 您应该为您的编码系统定义这些。 请注意，从技术上讲，编码类型是由媒体类型“限定”的(尽管我们尝试不将相同的四字符编码定义为两种媒体类型中的两种不同的编解码器，例如视频和音频，以避免混淆)。

### C.7.5 Sub-sample information

第12部分规范可以携带关于每个样本的“子样本”边界的信息。 然而，子样本的定义是特定于编码系统的。 在定义如何存储编码系统时，您可能希望定义它。

### C.7.6 Sample Dependency

第12部分的格式允许您识别编码系统的一些解码依赖关系信息。 特别是，您应该确定什么构成了有效的“同步”或随机访问点(可能从解码开始的点)。 它们可以在文件格式中进行标记(在同步示例表中，或者通过movie片段中的标记)。 如何标记同步样例应该不太值得关注。

同样，也可以指出哪些样本:

1. 依赖其他，或可独立解码;
2. 依赖其他，或可以在不影响解码的情况下丢弃;
3. 包含相同信息的多个编码，可能具有不同的依赖项(冗余编码)。

对于大多数编码系统，这些含义是不言而喻的，不需要拼写出来; 然而，对于某些编码系统，它们可能需要显式的语句。

### C.7.7 Sample Groups

样本群提供了另一种描述样本及其特征的方法。 要使用样例组，可以定义组类型，然后定义组的定义方式(组描述)。 然后，该文件格式可以将给定的样本映射到任意给定类型组的单个定义。 定义新的分组类型和参数化它们的方式是参数化文件格式的重要方法。

### C.7.8 Track-level

track可以通过两种重要的方式以文件格式相互关联。 Track引用是一个类型化的链接，表示一个track对另一个track的引用或依赖(例如，描述媒体track的meta-data track依赖于该媒体track，因为没有它就没有意义)。 新的track参考类型可以在派生规范中注册和使用。

类似地，track也可以被分组成可选的一组，读者可以从中选择适合自己的一组(例如，根据所支持的编解码器、比特率、屏幕大小等)。 3GPP 26.234采用了这个概念，并包含了用户数据(一个允许的扩展)，以提示为什么一个track是一个组的成员(“我包含一个不同的编解码器”)。

最后，可以在文件格式中启用或禁用track。 例如，禁用的track可以用于可选功能(如封闭的字幕)。

### C.7.9 Protection

与通过使用相同的输入类型和示例输入中的额外box来参数化编码方案类似，第12部分格式允许对track应用保护，由方案类型和scheme information box的内容进行参数化。 scheme information box由方案类型“拥有”——在某种程度上，其中包含的box不需要注册，因为它们已经被方案类型限定了范围。

保护可以是微妙的; 例如，许多加密系统都是“链”在一起的。 加密“mdat box的内容”是很诱人的，但这是非常严重的另人不适的对文件的微小更改。 保护块也很诱人——它们似乎确实代表了一个track上连续运行的媒体数据。 但是，再一次，re分块文件可能会破坏反保护的能力。

相反，可以考虑修改示例，或引入时间并行元数据，或使用示例组，以引入足够的上下文来支持基于文件的操作和解密。 时间并行元数据将在一个track中，应该使用一个track引用来指示受保护的数据依赖于并行加密上下文track。

## C.8 Construction of fragmented movies

在构造用于回放的片段文件时，有一些建议可以优化回放和随机访问的内容结构。 建议如下:

* 文件应由以下顺序的框组成:

- 'ftyp'

- 'moov'

- pair of 'moof' and 'mdat' (arbitrary number)

- 'mfra'

* 一个“moof” box 包含每个媒体最多一个“traf”。 当文件包含一个单独的视频track和一个单独的音频track时，'moof'将包含两个'traf'，一个用于视频，一个用于音频。
* 对于视频，随机可访问的样本被存储为每个“traf”的第一个样本。 在解码器逐步刷新的情况下，一个随机可访问的样本和相应的恢复点存储在同一个movie片段中。 对于音频，与每个视频随机可访问的样本有最接近的呈现时间的样本被存储为每个“traf”的第一个样本。 因此，在“moof”中，每种介质的第一个样本具有近似相等的呈现时间。
* 首先(随机可访问)样本被记录在视频和音频的“mfra”中。
* “mdat”中的所有样本都以适当的交错深度进行交错。

对于音频和视频，每个“moof”的偏移量和初始呈现时间在“mfra”中给出。

播放者将首先加载‘moov’和‘mfra’，并在播放期间将它们保存在内存中。 当需要随机访问时，播放者将搜索“mfra”以找到与指定时间最接近的随机访问点。

因为“moof”中的第一个样本是随机访问的，所以玩家可以在随机访问点上跳转目录。 玩家可以从一开始就读取随机访问点的“moof”。 随后的'mdat'从随机的可访问样本开始。 因此，两步搜索对于随机访问是没有必要的。

请注意' mfra' box 是可选的，并且可能永远不会出现在给定的文件中。

## C.9 Meta-data

上面提到的关于track及其数据的大部分内容都适用于元数据项，当然，元数据项没有时间结构。 特别地，将项划分为区段(允许它们交叉)也是文件格式的一个属性。基于区段结构设计新的支撑点是错误的。

## C.10 Registration

注册! 如有疑问，可通过http://www.mp4ra.org联系登记管理机构。 注册是免费的，建议和帮助也是免费的。 不注册意味着您的使用可能与他人冲突，您的使用也无法追溯，因此没有文档记录。 RA知道(至少)有许多brand是被愉快地发明和使用的，但却没有注册。 这些人在“挺而走险”; 不要加入他们的行列。

## C.11 Guidelines on the use of sample groups, timed metadata tracks,and sample auxiliary information

ISO Base Media File Format包含三种定时元数据机制，它们可以与特定的样本相关联:样本组、定时元数据track和样本辅助信息。 派生规范可以使用这三种机制中的一个或多个提供类似的功能。 这个子句为派生规范提供了在这三种机制之间进行选择的指导方针。

样本组和定时元数据与媒体数据的耦合不那么紧密，通常是“描述性的”，而解码可能需要样本辅助信息。

示例辅助信息仅用于与示例直接相关的一对一的情况，并用于媒体示例的处理和展示。 对于一般内容，应该使用现有的附加音轨解决方案。 样例辅助信息和样例媒体数据都是使用字节指针和大小信息来寻址的，因此，当相同字节构成多个样例的数据时，可以通过重用相同的字节指针来共享这些数据。

样本组在下列情况下可能有用。

* 当多个样例共享相同的元数据值时，在Sample Group Description box中指定元数据以及在Sample to Group box(es)中指定样例到元数据的关联是节省空间的。
* 由于样例组信息存储在Movie box和Movie Fragment box(es)中，它们为Media Data boxes中的数据提供了索引。 与定时的元数据跟踪和样例辅助信息相比，不需要从Media Data boxes中获取数据，因此可以减少磁盘访问。

定时元数据track在以下情况下可能有用。

* 同一个计时元数据track可以关联到多个track。 换句话说，定时元数据track可能比样本组和样本辅助信息更独立于相关track的内容。
* 跟样本辅助信息或元数据跟踪样本组相比，它可能更容易添加一个文件的时间，因为样品辅助信息和Sample to Group boxes必须驻留在相同的track碎片里作为相关的样本,而时间元数据可能驻留在自己的Movie Fragment box(es)。 例如，与使用示例辅助信息相比，提供额外的字幕track作为计时元数据可能更容易。
* 定时元数据样本的持续时间不需要与关联媒体或提示样本的持续时间匹配。 如果定时元数据样本的持续时间跨越多个相关媒体或提示样本，那么定时元数据track可能比样本辅助信息更节省空间。

示例辅助信息在以下情况下可能有用。

* 与样本相关的数据变化非常频繁，因此从存储空间的角度来看，指定样本组可能是不合理的。
* 与样本相关联的数据量是如此之大，以至于在Movie box或Movie Fragment box(根据样本分组的要求)中携带数据会造成不利影响。 例如，在渐进式下载中，为了保持较小的初始缓冲时间，使Movie box的大小较小可能是有益的。
* 当每个示例都与元数据相关联时，与使用定时元数据track的相同功能相比，示例辅助信息提供了辅助信息与示例之间更直接的关联。 这通常需要解析样本解码时间来建立定时元数据样本和媒体/hint样本之间的关联。

# Annex D

(informative)

**Registration Authority**

## D.1 Code points to be registered

文件格式中的代码点都是32位字段，通常是四个可打印字符(通常称为四字符码或4CCs)。 objecttype

标识符是一个8位整数。

可注册的代码点(code-point)如下:

1. 文件格式框标识符。 注意，在某些规范中，box被称为原子。 注意，不鼓励引入新的原子类型; 一般来说，如果可能的话，应该使用文件格式的其他可扩展特性。
2. 文件格式 track 类型标识符。 这里通常使用一对标识符来标识 track 类型(音频、视频等)，如果需要，还可以使用特定于媒体的报头原子(视频媒体报头等)。 然而，预计对新 track 类型的需求很少; 大多数媒体应该属于现有的类型(例如，视频编解码器应该使用视频 track ，hint 协议使用 hint track，等等)。
3. 文件格式示例描述和示例格式标识符(也称为编解码器名称)。 这包括音频和视频编解码器，以及 hint track 的协议标识符。 任何新示例格式的注册都将自动获得一个对象类型标识符(见下文)，从而使在MPEG-4系统对象描述符框架中识别这种格式的承载成为可能。
4. 文件格式track引用标识符。 track之间的依赖关系以文件格式输入(例如，hint track依赖于它们hint的媒体track，使用类型为“hint”的track依赖关系)。
5. 该规范包括一个“文件类型”原子，该原子包含一个 “brand” 列表，用于标识该文件符合哪些规范。 根据这种文件格式的结构定义定义标准的机构通常会使用一个新的 brand 来标识符合其规范的文件。 任何新 sbrand 的注册都必须注明该 brand 识别的准确规格。
6. 在MPEG-4对象描述符框架中，对象类型值用于标识流的格式。 可以独立于上面的文件格式标识符请求对象类型标识符。
7. 样本组将类型化信息与样本组相关联。 分组类型可以注册。
8. 媒体和元数据都可以被保护，并且保护方案使用注册的保护方案类型来标识。

这些代码点(code-point)在本附件的其余部分中称为注册标识符，缩写为RID。

## D.2 Procedure for the request of an MPEG-4 registered identifier value

MPEG-4代码点(code-point)的请求者，如上述值所详细说明，以识别私人数据格式，应向注册机构申请。 登记表格由登记机关提供。 请求者应提供D.4中规定的信息。 公司和组织有资格申请。

## D.3 Responsibilities of the Registration Authority

本附件概述了登记机构管理私人数据格式标识符登记的主要责任; 某些其他职责可以在JTC 1指令中找到。 登记机关应当:

1. 按照JTC 1指令，实施申请唯一RID的注册程序;
2. 从应用程序提供者接收和处理用于分配标识符的应用程序;
3. 确定哪些申请符合本登记程序，并在收到申请后30天内通知申请人已分配的RID;
4. 如申请被拒绝，应在收到申请后30天内以书面通知申请提供者，并考虑及时重新提交申请;
5. 维护已分配标识符的准确寄存器。 对格式规范的修改，应当由登记主管机关接受并保存;
6. 应要求，将本登记册的内容提供给国际标准化组织(ISO)或国际电工委员会(IEC)成员的JTC 1国家机构，以及国际标准化组织(ISO)或国际电工委员会(IEC)的联络组织和任何相关方;
7. 维护一个RID请求表单的数据库，允许和拒绝。 寻求有关具有RID的私人数据格式的技术信息的各方应有权访问这些信息，这些信息是登记机构维护的数据库的一部分;
8. 每年向JTC 1、ITTF和SC 29秘书处或其各自指定的机构报告其活动; 和
9. 尽可能适应现有RID的使用。

## D.4 Contact information for the Registration Authority

Apple Computer Inc.

One Infinite Loop, M/S 301-4B

Cupertino, California 95014

USA

E-mail: mp4reg@group.apple.com

Web: http://www.mp4ra.org/

## D.5 Responsibilities of Parties Requesting a RID

要求格式标识符的一方应:

1. 使用登记机关提供的表格和程序申请;
2. 在申请表格内注明注册识别码的用途，以及所需的技术资料;
3. 提供联系信息，说明如何在非歧视性的基础上获得完整的描述;
4. 同意在合理时间内制定已授予的RID的预期使用情况; 和
5. 保存申请表和从登记机构收到的已授予RID的通知的永久记录。

## D.6 Appeal Procedure for Denied Applications

注册管理小组对被拒绝的RID申请的上诉有管辖权。 RMG的成员应由负责ISO/IEC 14496的ISO技术委员会的P-和L-成员提名。 委员会应有一名召集人和从其成员中提名的秘书处。 登记机关有权提名一名无表决权的观察成员。

RMG的职责应为:

1. 在合理时间范围内审查和处理所有上诉;
2. 以书面通知提出上诉的组织重新考虑其关于RMGs对该事项的处理的请求;
3. 审查登记机关年度活动总结报告; 和
4. 向ISO成员机构和IEC国家委员会提供关于注册机构的运作范围的信息。

## D.7 Registration Application Form

**D.7.1 Contact Information of organization requesting a RID**

Organization Name:

Address:

Telephone:

Fax:

E-mail:

Telex:

**D.7.2 Request for a specific RID**

—如果本系统已实现并已使用，请填写此项和D.7.3，并跳转到D.7.5;否则请留空，并跳转到D.7.3)

**D.7.3 Short description of RID that is in use and date system was implemented**

**D.7.4 Statement of an intention to apply the assigned RID**

**D.7.5 Date of intended implementation of the RID**

**D.7.6 Authorized representative**

Name:

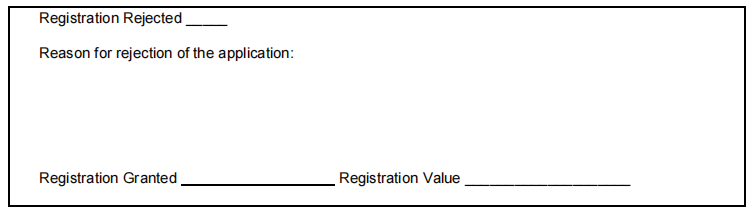
Title:

Address:

Email:

Signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**D.7.7 For official use of the Registration Authority**



Attachment 1 ⎯ Attachment of technical details of the registered data format.

Attachment 2 ⎯ Attachment of notification of appeal procedure for rejected applications.

# Annex E

(normative)

**File format brands**

E.1 Introduction

ftyp框的compatible\_brands列表中出现的brand是声明和权限。 声明文件符合该brand的所有要求，并允许可能只实现该brand的读取器读取该文件。

一般情况下，读者被要求实现一个品牌的所有功能文档，除非以下其中之一适用:

1. 他们正在使用的媒体不使用或需要一个特性:例如，i帧视频不需要一个同步样本表，如果合成重新排序不使用，那么合成时间偏移表是不需要的; 类似地，如果不需要内容保护，则不需要支持内容保护的结构。
2. 该文件所遵循的另一个规范禁止使用某个特性(例如，一些派生的规范明确禁止使用movie片段);
3. 产品运行的环境意味着某些结构是不相关的; 例如，hint track结构只与为hint track中的协议准备内容或执行文件交付(例如流)的产品相关。

以下部分列出了本规范中定义的brand; section顺序没有隐含任何继承——当发生继承时，它是特别声明的。 其他brand可在其他规范中定义。 注意，如果一个brand是另一个brand的子集(例如，' isom '要求是' iso2 '要求的子集)，那么:

1. 被标记为与子集兼容的文件总是可以被标记为也与超集兼容; 一个兼容' isom '的文件总是可以被标记为兼容'iso2';
2. 自动支持超集的产品可以支持子集; 支持' iso2 '的产品也必须支持' isom '。

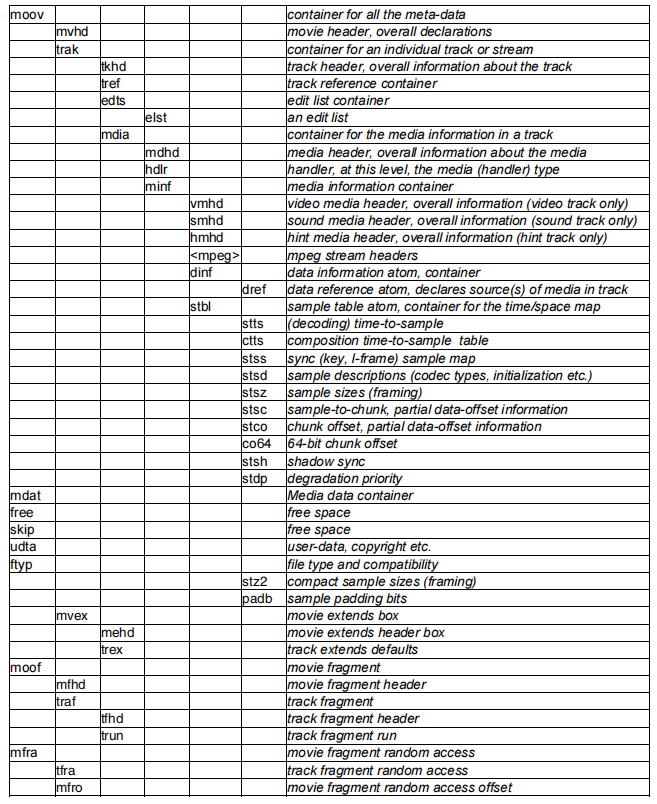
这里定义的任何brand都不需要支持任何特定的媒体类型(例如，视频、音频、元数据)或媒体编码(例如，特定的编解码器)，或支持特定媒体类型的结构(例如，视觉样本条目或包含在特定类型的样本条目中的box)。

可以使用更具体的标识符来标识提供更详细信息的规范的精确版本。 这些brand不应作为主要brand; 这个基本文件格式应该派生成要使用的另一个规范。 因此，没有定义的普通文件扩展名或指定给这些brand的mime类型，也没有定义当其中一个brand是主要brand时的小版本。

## E.2 The ‘isom’ brand

类型' isom ' (ISO基础媒体文件)在本规范的本节中被定义为符合ISO基础媒体文件格式第一版的标识文件。

需要支撑以下结构的box:



Hint track必须被识别，在hint track中，RTP协议hint track。

请注意，Track Header Box的一些要求不适用于这个brand; 看到副条款8.3.2.1。

此处只需要支持' ctts ' box的版本0; 版本1支持不是必需的。

此处只需要对' turn ' box的版本0进行支持; 版本1支持不是必需的。

说明default-base-is-moof标志(8.8.7.1)不能设置在有此标志的文件上。

## E.3 The ‘avc1’ brand

Brand “avc1”应使用，以表明该文件符合第8.6.4和8.9条中的“AVC扩展”。 如果不使用其他brand，这意味着需要对这些扩展的支持。 使用“avc1”作为主要brand可能是规范所允许的; 在这种情况下，该规范定义了文件扩展名和所需的行为。

“avc1” brand 需要对“isom” brand的支持。 此外，还需要支持下列各栏:



在示例组中，需要支持滚动组(分组类型为“roll”)。

注意default-base-is-moof标志(8.8.7.1)不能设置在有此标志的文件上

请注意，Track Header Box的一些要求不适用于这个brand; 看到副条款8.3.2.1。

此处只需要支持' ctts '框的版本0; 版本1支持不是必需的。

此处只需要对' turn ' box的版本0进行支持; 版本1支持不是必需的。

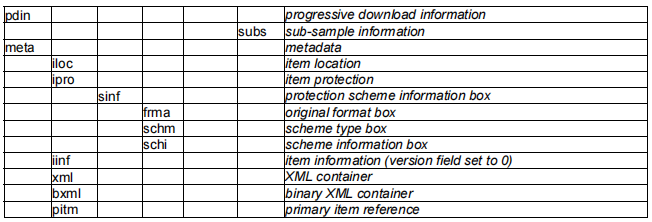
不需要支持movie片段中的SampleGroupDescription box。

## E.4 The ‘iso2’ brand

Brand ' iso2 '应使用表示兼容ISO基础媒体文件格式第二版; 它可以用于“isom” brand 的补充或替代，且适用相同的使用规则。 如果使用时没有标明本规范第一版的brand “isom”，则表明需要支持第8.6.4、8.9、8.11.1至8.11.7、8.11.10、8.12小节中的部分或全部技术，或第9.1小节中的SRTP支持。

' iso2 ' brand需要支持' avc1 '品牌的所有特性。

此外，还需要对下列box提供支持:



在RTP hint track的上下文中，现在必须识别SRTP提示跟踪。 需要内容保护和广义meta-data boxes支持。

只需要支持item information box的版本0和item location box的版本0。

请注意，Track Header Box的一些要求不适用于这个brand; 看到副条款8.3.2.1。

此处只需要支持' ctts ' box 的版本0; 版本1支持不是必需的。

此处只需要对' turn ' box的版本0进行支持; 版本1支持不是必需的。

不需要支持movie片段中的SampleGroupDescription box。

请注意default-base-is-moof标志(8.8.7.1)不能设置在有此标志的文件上。

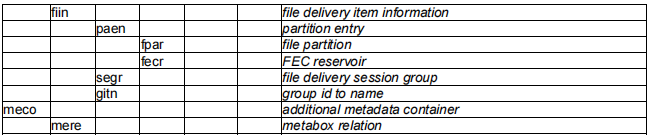
## E.5 The ‘mp71’ brand

如果一个具有MPEG-7处理程序类型的Meta-box在文件级别被使用，那么 brand ' mp71 '应该是file-type box中兼容brand列表的成员。

## E.6 The ‘iso3’ brand

“iso3” brand需要支持“iso2” brand 的所有特性。

此外，还需要对以下方面提供支持:



需要支持项目信息框的版本0和版本1。 在样本组中，需要支持速率共享信息(分组类型为' rash ')。 文件传递hint track (示例条目' fdp ')必须被识别。

此处只需要支持' ctts ' box的版本0; 版本1支持不是必需的。

此处只需要对' turn ' box的版本0进行支持; 版本1支持不是必需的。

不需要支持movie片段中的SampleGroupDescription box。

只需要支持item location box的版本0。

请注意default-base-is-moof标志(8.8.7.1)不能设置在有此标志的文件上。

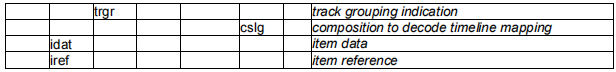
## E.7 The ‘iso4’ brand

“iso4” brand需要支持“iso3” brand的所有特性。

该品牌下需要支持版本1的组合偏移(' ctts '和' iloc ') box。

需要支持版本1的item location box，版本2的item info box以及新的item data box(' idat ')和item reference box (' iref ')。

此外，还需要对以下方面提供支持:



此处只需要对' turn ' box的版本0进行支持; 版本1支持不是必需的。

不需要支持影片片段中的SampleGroupDescription box。

请注意 default-base-is-moof标志(8.8.7.1)不能设置在有此标志的文件上。

## E.8 The ‘iso5’ brand

“iso5” brand 需要支持“iso4” brand 的所有特性。

该brand下需要支持default-base-is-moof标志。

在本brand下，需要处理受限制的样本条目(即“resv”)。

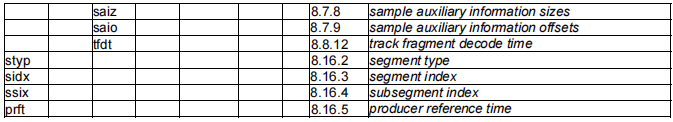
此处只需要对' turn ' box的版本0进行支持; 版本1支持不是必需的。

不需要支持movie片段中的SampleGroupDescription box。

## E.9 The ‘iso6’ brand

“iso6” brand 需要支持“iso5” brand 的所有特性。

本brand下需要支持以下box:



本brand需要以下支持:

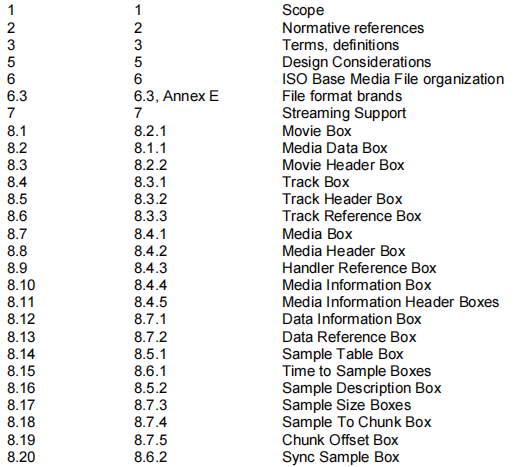
* movie片段中的描述box;
* track run boxes(即track run boxes的版本1)中的有符号组成偏移量;
* 在示例组中，需要支持随机访问点信息(分组类型为“rap”)。

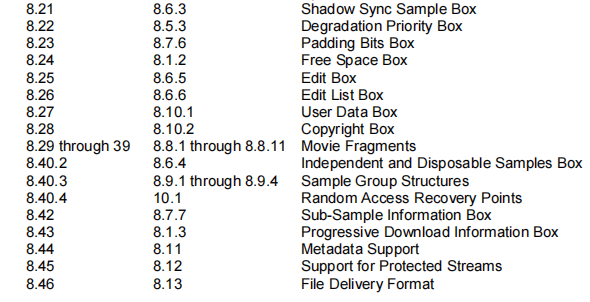
# Annex F

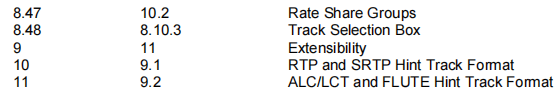
(informative)

**Document Cross-Reference**

在第二版出版后，本国际标准被重新组织，先前的第8条(“box结构”)被重新组织，随后的条款被重新组织，相应的章节编号也被重新组织。 为了能够遵循在本国际标准的前一个版本中有章节编号的引用，可以使用下表。







# Annex G

(informative)

**URI-labelled metadata forms**

## G.1 UUID-labelled metadata

UUID标记元数据的URI格式在IETF RFC 4122中定义:统一唯一标识符(UUID) URN命名空间(2005年7月)。

没有关于主元数据、时态元数据的初始化数据或时态元数据本身形式的通用语句。 所有这些的形式都取决于精确的UUID及其定义。

请注意，uuid不能很容易地追踪到它们的起源点，因此，如果希望元数据的接收者能够在需要时找到相关的文档，那么uuid可能不合适。

如果需要可追溯性，则应该使用标准化的元数据框架(如MPEG-7)或注册的框架(如SMPTE)或可取消引用的URL。

## G.2 ISO OID-labelled metadata

带有oid标签的元数据的URI格式在RFC 3061:对象标识符的URN名称空间(2001年2月)中定义。

没有关于主元数据、时态元数据的初始化数据或时态元数据本身形式的通用语句。 所有这些的形式都取决于精确的对象标识符及其定义。

许多更具体的标签系统也可以表示为对象标识符。 应该使用更具体的UUID形式。

对象标识符开始{joint-iso-itu(2) uuid(25)}(即开始urn:oid:2.25)不应该被使用; UUID uri应该直接使用。

从{iso(1) identified-organizations(3) SMPTE(52) metadata-dictionary(1)}(即urn:oid:1.3.52.1)开始的对象标识符不应该被使用，也不应该使用任何其他oid作为根据SMPTE 298M或336M的标签; 应该使用更具体的SMPTE URI形式。

对象标识符注册到特定的组织，因此可以标识拥有特定标识符的组织。 但是，对象标识符树的某些部分被委托给未注册的用处(如上面提到的uuid)，这样就失去了可跟踪性。

如果需要可追溯性，则应该使用标准化的元数据框架(如MPEG-7)或注册的框架(如SMPTE)或可取消引用的URL。

## G.3 SMPTE-labelled metadata

smete标记元数据的URI格式在RFC 5119中; Society of Motion Picture and Television Engineers(SMPTE)的Uniform Resource Name(URN)命名空间。

主元数据正是SMPTE 336M中定义的KLV(键、长度、值)三元组的值(V)部分，键是URN中给出的标签，长度(L)来自项的长度。

类似地，每个时态元数据样本都是KLV的值(V)部分，其中关键是匹配样本条目中给出的URN标签，长度(L)来自样本大小(在样本大小或紧凑的样本大小表中给出)。

可能存在初始化数据。 它包含KLV的键(K)和值(V)，为由样本组成的KLV提供初始化上下文，长度(L)来自DataBox大小。 前16个字节是初始化数据的SMPTE标签，按SMPTE 336M中定义的方式存储，然后是数据。

这些字节的典型值，如SMPTE 377M中定义的，是‘primer pack’(十六进制):06 0E 2B 34 02 05 01 0D 01 02 01 01 01 00。 如果初始化数据的标签实际上没有标识出提供上下文信息的结构(例如primer pack)，则该行为是未定义的。 这使得每个样本都是一个本地集。 必须遵守SMPTE 377M中定义的本地集构造规则。

SMPTE 377M使用定位器来定位元数据本身之外的其他资源。 对于静态元数据，它们应该使用meta-box中的item location box。 对于时态元数据，可以直接使用外部指针。

初始化数据可能不存在，然后标签识别不需要上下文的特定元数据项(例如地理定位器)。

# Annex H

(informative)

**Processing of RTP streams and reception hint tracks**

## H.1 Introduction

### H.1.1 Overview

本附件提供了RTP流的记录建议，以及使用记录的RTP流进行回放和重新发送。

### H.1.2 Structure

本附件的组织结构如下:

* H.2介绍了RTP流回放可能变得不同步的潜在原因，并概述了如何在记录和回放中促进正确的同步。 它在其他条款之前，因为录音设备和播放器都必须采取行动来实现适当的同步。
* H.3提供了存储RTP流的建议。
* H.4提供了如何播放包含录制RTP流的文件的建议。
* H.5为重新发送H.3中描述的存储在文件中的接收RTP流提供了建议

### H.1.3 Terms and definitions

为本附件的目的，适用下列术语和定义。

#### H.1.3.1 player

实体，它解析文件、解码文件中至少一个tracck子集，并呈现解码后的track。

#### H.1.3.2 recording unit

一种实体，它接收一个或多个封装和压缩的媒体包流，并将接收到的媒体存储到一个文件中。

#### H.1.3.3 re-sending unit

解析包含媒体的文件的实体，该媒体源自封装和压缩的媒体的一个或多个接收的包流，并传输存储在该文件中的媒体的至少一个子集。

## H.2 Synchronization of RTP streams

对于接收到的RTP流，有几个潜在的非同步回放来源。当RTP流被记录为RTP接收hint track时，还记录了保证同步回放的必要信息。当RTP流被记录为媒体track时，必须通过适当地创建媒体样本的合成时间来保证媒体track的回放的同步。下面的列表描述了对接收到的RTP流的非同步回放的来源，总结了推荐的同步方法，并指向相关的条款以获取进一步的信息。

流的第一个包的RTP时间戳有一个随机偏移。 因此，两个流的RTP 时间戳会进行移位，这个移位通过它们的初始随机偏移值进行，即使不同流的RTP时间戳的可能的差异时钟速率（potentially different clock rate）被补偿。 随机偏移量应该反映在H.3.5中所述的接收hint样本条目的'tsro' box的偏移量字段的值中。

不同流的第一个接收和记录数据包可能不具有H.3.2中讨论的相同的回放时间。 不同记录流的不相等的开始时间可以通过解析一个或多个RTCP发送者报告来得到播放时间作为发送者的挂钟时间，并使用H.3.2中描述的Edit List box创建播放的初始偏移量来补偿。Edit List box由播放者解释，如H.4.4所述。

不能保证生成某个RTP流的RTP时间戳的时钟与发送者的挂钟时间以相同的运行速度运行，后者用于创建RTCP发送者报告。例如，RTP时间戳可以根据恒定的采样频率生成，例如音频为44.1kHz，因此由音频捕获硬件的时钟速率控制。然而，RTCP发送者报告可以根据与音频捕获硬件的时钟不同的时钟运行的系统时钟来生成。此外，用于为音频生成RTP时间戳的时钟可能以不同于用于为视频生成RTP时戳的时钟的速度运行（当两者归一化为相同的时钟滴答频率时）。

在播放者中也会出现类似的问题，如果为已解码流的输出设定节奏的时钟（clock pacing）与玩家的挂钟的节奏不同，或者为不同已解码流的呈现设定节奏的时钟没有同步。

对于所有这些以不同速度运行的时钟的潜在问题，推荐的方法是使用RTCP发送者报告将不同流的RTP时间戳对齐到相同的墙上时钟时间轴上，这用于流间同步。这种对齐可以通过修改记录的RTP时间戳的表示来记录流，或者通过使用H.3.6中描述的记录的RTCP发送者报告来播放录制的流。此外，建议根据H.4.4中所述的音频播放速率来调整播放速度。

发送方的挂钟可能以不同的挂钟运行。

建议以播放器的挂钟的速度播放一个录制的程序，并使用音频播放时钟作为播放器的挂钟。因此，音频时间尺度。即使播放者的壁钟与发送者的壁钟运行的速度不同，它通常也是不被注意到的。

在H.4.4中描述了解码介质样本输出的间隔。

## H.3 Recording of RTP streams

### H.3.1 Introduction

RTP流的记录可以产生三个基本的文件结构。

1. 一个只包含RTP接收hint track的文件。不包括任何媒体曲目。这种文件结构能够有效地处理数据包丢失，但只有能够解析RTP接收hint track的播放器才能播放该文件。
2. 一个只包含媒体跟踪的文件。不包括RTP接收hint track。这种文件结构允许现有的播放器兼容的早期版本的ISO基础媒体文件格式处理记录的文件，只要媒体格式也被支持。然而，由于后续子句中解释的原因，复杂的传输错误处理是不可能的。
3. 一个同时包含RTP接收提示跟踪和媒体跟踪的文件。该文件结构具有上面提到的好处，当与来自ISO基础媒体文件格式的其他文件格式具有尽可能良好的互操作性时，应该使用。

如果被记录的RTP流受到保护，则使用受保护的RTP接收hint track代替RTP接收hint track，否则记录单元的操作保持不变。在回放时，受保护的RTP接收hint track中包含的数据首先不受保护，然后与传统的不受保护的RTP流进行处理。或者，在将RTP流存储为RTP接收hint track之前可能不受保护，但是必须注意遵守使用受保护RTP流中内容的权利。

一些记录操作对于所有这三种文件结构都是常见的，而其他一些则不同。表H.1指出了基本文件结构需要哪些记录操作。

**Table H.1**

|  |  |  |  |
| --- | --- | --- | --- |
|  | File containing only  RTP reception hint  tracks | File containing only  media tracks | File containing both  RTP reception hint  tracks and media tracks |
| Compensation for  unequal starting position  of received RTP  streams  (H.3.2) | no, when RTCP  reception hint tracks  are stored;  yes, otherwise | yes | no, when RTCP  reception hint tracks  are stored;  yes, otherwise |
| Recording of SDP  (H.3.3) | yes | no | yes, for RTP reception  hint tracks only |
| Creation of a sample  within an RTP reception  hint track (H.3.4) | yes | no | yes, for RTP reception  hint tracks only |
| Representation of RTP  timestamps  (H.3.5) | yes | no | yes, for RTP reception  hint tracks only |
| Recording operations to  facilitate inter-stream  synchronization in  playback  (H.3.6) | yes | yes, the composition  times of media tracks  should be compensated  as described in H.3.6.3 | yes |
| Representation of  reception times  (H.3.7) | yes | no | yes, for RTP reception  hint tracks only |
| Creation of media  samples  (H.3.8) | no | yes | yes, for media tracks  only |
| Creation of hint samples  referring to media  samples  (H.3.9) | no | no | yes |

有些实现可能首先只记录RTP接收hint track，并创建一个结合了媒体跟踪和RTP接收hint track的文件。

### H.3.2 Compensation for unequal starting for position of received RTP streams

当对RTP流的记录开始时，至少由于以下原因，一个RTP流中的第一个媒体样本的呈现时间不等于另一个RTP流中的第一个媒体样本的呈现时间：

* 音频和视频的采样频率通常会有所不同。
* 音频和视频流可能在传输顺序中的呈现时间方面不完全交错。

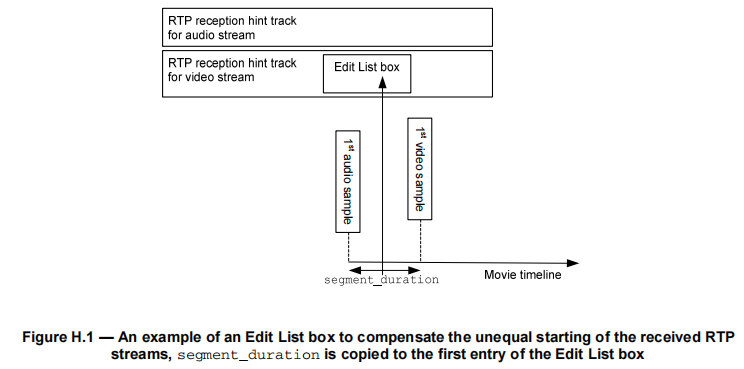
如果存储RTCP接收hint track，则应在播放时对接收RTP流的起始位置进行补偿，不创建有关RTP接收提示轨迹的Edit List box。如果没有存储RTCP接收提示轨迹，或者如果存储了媒体轨迹，则记录单元必须指示流的相对初始延迟，以便在本条款中随后描述的流的回放开始时正确地同步音频和视频。记录单元应执行以下操作。

1. RTCP发件人报告指示哪个RTP时间戳对应于发送报告即时时间的壁纸时间。至少应该解析每个RTP流的第一个RTCP发送者报告，以建立每个RTP流的RTP时间戳和发送者的挂钟时间的等价性。通过简单的线性外推，为每个RTP流推导出最早接收到的RTP数据包的墙时钟时间戳。
2. 在所有接收的RTP流中导出的最小的挂钟时间戳映射到电影时间轴中的呈现时间戳0，即在记录文件的回放开始时立即呈现。影片时间轴是播放文件的主时间轴。
3. 每个轨道的媒体时间轴从0开始。为了将媒体时间轴移到电影时间轴中的正确起始位置，将为其他RTP轨迹（不包含具有最早壁纸时间戳的数据包）创建Edit box和Edit List box，如下所示：

Edit List box包含两个条目：

1. 第一个条目是一个空的编辑(由media\_time等于-1表示)，其持续时间(segment\_duration)等于所有RTP流中最早媒体样本和track中最早媒体样本的呈现时间之差。图H.1给出了一个如何导出Edit List box中第一个条目的segment\_duration的示例。
2. 第二条目的media\_time值等于按表示顺序排列的最早样本的组成时间，第二条目的segment\_duration值跨越整个track。由于创建Edit List box时可能不知道实际跟踪持续时间，建议设置segment\_duration等于最大可能值（最大32位无符号整数或最大64位无符号整数，具体取决于使用的box的版本）。

在Edit List box的两个条目中，media\_rate\_integer的值均等于1。



一些记录单元可以检测可以从其中开始解码的数据包，例如H.264/AVC流的IDR图片，这在这里被称为随机接入点。如果流包含在所有接收流中具有最早的墙时钟时间戳的包，并且同一流包含以解码顺序在流的第一随机接入点之前的包，则建议不要存储在流的第一随机接入点之前的包，并且在确定所有接收流中最早的墙时钟时间戳时不考虑它们。

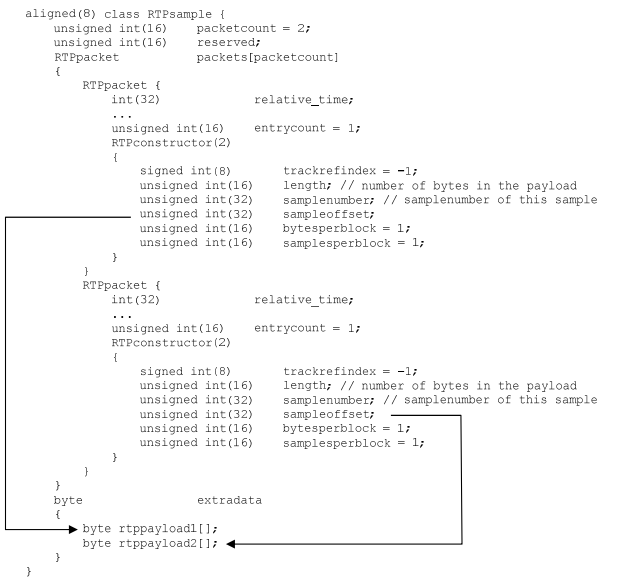
### H.3.3 Recording of SDP

SDP应存储如下。会话级SDP，即第一个媒体特定行(“m=”行)之前的所有行，应作为影片SDP信息存储在User Data box中，如9.1.4.1所述。SDP描述中的每个媒体级部分都以“m=”行开始，然后继续到下一个媒体级部分或整个会话描述的结束。每个媒体级部分应作为track SDP信息存储在相应的RTP接收hint track的User Data box中。

### H.3.4 Creation of a sample within an RTP reception hint track

建议每个示例代表所有接收到的具有相同RTP时间戳的RTP数据包，即具有公共RTP时间戳的RTP序号顺序的连续数据包。RTPsample结构设置为每个接收的具有相同RTP时间戳的RTP包包含一个RTP包结构。建议每个RTP包包含一个类型为2的包构造函数(RTP样本构造器)。RTP采样构造器复制由构造函数的样本偏移和长度字段表示的特定字节范围，由构造函数的采样数字段表示，通过引用到正在构造的包有效负载。每个接收到的具有相同RTP时间戳的RTP数据包的有效负载被复制到示例的外部数据部分。每个构造函数的track引用被设置为指向hint track本身，即被设置为-1，样本偏移和长度被设置为匹配样本中数据包有效负载的位置和大小。

图H.2给出了一个RTP接收提示示例的伪代码示例，其中包含两个RTP数据包。



**Figure H.2 — An example of a RTP reception hint sample containing two packets**

**(their header and payload)**

不建议使用错误发生索引事件来指示RTP包丢失，因为RTPsequenceseed种子字段可以用于检测包丢失，而不增加存储空间。此外，错误发生事件可以引用的最小单位是一个样本(在RTP接收hint track中)。由于一个示例可以包含许多数据包，所以错误发生索引事件所涉及的哪个数据包是不明确的。

### H.3.5 Representation of RTP timestamps

RTP时间戳在RTP接收hint track中用三个值的和表示，其中一个是track的媒体时间轴中的解码时间DT。解码时间将运行长度编码到Decoding Time to Sample box中，并另外编码到一个或多个Track Fragment Run boxes中。Decoding Time to Sample box 包括若干个sample\_count和sample\_delta对，其中sample\_delta是一组连续样本中每个样本的解码时间增量（即，在解码时间中的样本持续时间），其数量等于sample\_count。Track Fragment Run box表示一对sample\_count和sample\_duration，其中sample\_duration是一组连续样本中每个样本的解码时间增量（即样本持续时间），其数量等于sample\_count。每个Track Fragment box可以包含多个Track Fragment Run boxes。样本数i的解码时间DT(i)是通过将Decoding Time to Sample box中样本i之前的所有样本的样本持续时间相加，如果需要，还可以引用样本i之前的任何样本i的Track Fragment Run boxes。

样本i的RTP时间戳RTPTS(i)由如下指定的三个值的和表示：

RTPTS(i) = (DT(i) + tsro.offset + offset) mod (H.1)

其中，tsro.偏移是引用接收hint样本输入的“tsro” box中的偏移值的偏移值，偏移是RTP包结构中rtpoffsetTLV box中包含的值，mod是模操作。

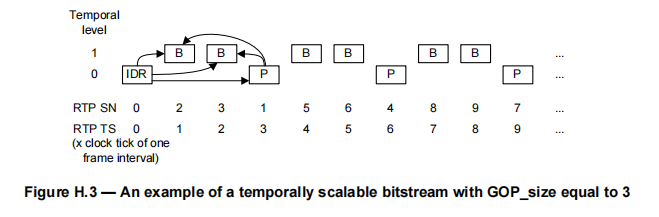
一个“tsro” box 应该出现在RTP接收hint示例条目中。track的任何“tsro” box中的偏移值应该以RTP序列号顺序的各自流的第一个包的RTP时间戳。

如果在样本i-1和i之间没有发生最大32位无符号整数的RTP时间戳值的环绕，则以RTP序号顺序排列的连续不等RTP时间戳之间的差为

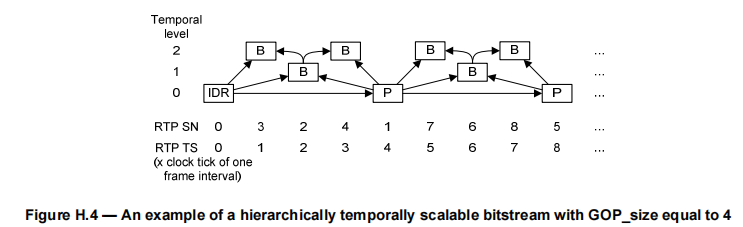
RTPTS\_DIFF(i) = RTPTS(i) – RTPTS(i – 1) for any i > 1 (H.2)

RTPTS\_DIFF(i)保持不变，当帧速率为恒定时，任何数据包中的帧数都是恒定的，并且传输顺序与呈现顺序相同。这些约束通常由音频流和暂时不可伸缩的视频流来满足。如果RTPTS\_DIFF(i)是一个常量，并用RTPTS\_DIFF表示，则建议使用以下方法。Decoding Time to Sample box中的sample\_delta值，如果使用movie片段，则将Track Fragment Run box(boxes)中的sample\_duration值设置为RTPTS\_DIFF，这将导致Decoding Time to Sample box和Track Fragment Run boxes的紧凑。如果使用RTCP接收hint track，则不应使用rtpoffsetTLV box(见H.3.6)。否则(如果不使用RTCP接收提示轨迹)，rtpoffsetTLV box中的偏移量应设置为0。

当在视频流中使用时间可伸缩性时，数据包的传输顺序和回放顺序不相同，RTP时间戳不会随着RTP序列号的函数而增加，而RTPTS\_DIFF(i)也不是常数。然而，RTP时间戳通常在由GOP\_size决定的周期内具有恒定的行为，这是1加上以RTP序列数顺序在最低时间水平上的两个连续图片之间的图片数量。例如，如果如图H.3所示，为每对参考图片编码两个非参考图片，则GOP\_size等于3。图H.4给出了一个GOP\_size为4的分层时间可伸缩位流的示例。



(RTP序列号(SN)被归一化，从0开始，并且假设每帧有一个数据包。RTP时间戳(TS)被标准化为从0开始，并表示为持续一帧间隔的时钟刻度。内部预测箭头只表示第一个共和党，而其他共和党中的图片也同样被预测。)

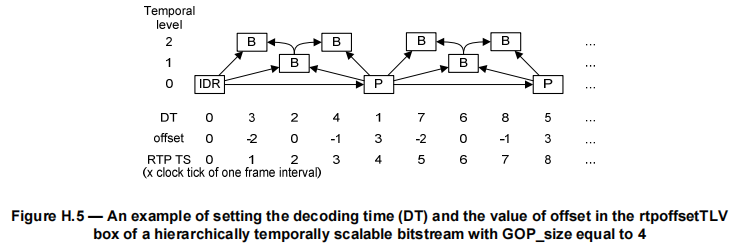


(RTP序列号(SN)被归一化，从0开始，并且假设每帧有一个数据包。RTP时间戳(TS)被标准化为从0开始，并表示为持续一帧间隔的时钟滴答。)

当样本i和i+GOP\_size之间没有环绕超过最大32位无符号整数的RTP时间戳值时，由一个GOP引起的RTP时间戳增量如下，包括：

RTPTS\_GOP\_DIFF(i) = RTPTS(i + GOP\_size) – RTPTS(i) (H.3)

如果RTPTS\_GOP\_DIFF(i)是一个常数等于RTPTS\_GOP\_DIFF，当没有样本i，i+1，...，i+GOP\_size时是开始所谓的封闭一组图片的图片，例如H.264/AVC流的IDR图片，推荐以下内容。Decoding Time to Sample Box中的sample\_delta值，如果使用movie片段，则Track Fragment Run box(es)中的sample\_duration值设置为RTPTS\_GOP\_DIFF/GOP\_size。如果使用了RTCP接收hint track，则对于最低时间水平的图片，不应使用该box(见H.3.6)。否则(如果不使用RTCP接收hint track)，rtpoffsetTLV box中的偏移量应设置为0。对于其他时间级别的图片，应该设置rtpoffsetTLV box中的偏移值，以满足公式H.（1）。图H.5显示了如何为图H.4中所示的分层时间可伸缩的视频位流设置解码时间和偏移量。



(在本例中，将样本之间的解码时间增量设置为RTPTS\_GOP\_DIFF/GOP\_size，以具有紧凑的编码解码时间。对每个样本的rtpoffsetTLV box中的偏移量值进行调整，以存储RTP时间戳的表示。对于这个例子，RTP时间戳和解码时间被标准化为从0开始，并表示为持续一帧间隔的时钟刻度。)

如果从接收到的包中没有检测到RTP时间戳的线性和周期性行为，并且两个接收到的不同样本的包没有相同的接收时间，建议在Decoding Time to Sample Box中设置sample\_delta的值，如果使用movie片段，则使用Track Fragment Run box(es)中sample\_duration的值来表示样本的第一个包的接收时间。即，导出的解码时间DT(i)应等于样本的第一包的接收时间减去流的第一接收样本的第一包的接收时间。

需要注意的是，文件中没有在任何hint track中的样本中明确表示组合时间戳。因此，对于RTP接收hint track，合成时间戳从存储的包流中指示的与RTP时间戳相关的信息推断出合成时间戳。对于与RTCP接收hint track没有关联的RTP接收hint track，推断接收的RTP包的合成时间是采样时间DT(i)和包括样本的rtpoffsetTLV box中偏移字段的值的和。对于与RTCP接收hint track相关联的RTP接收hint track，合成时间推断如下。让在同一轨道内具有最早RTP时间戳的接收RTP包的合成时间等于0。任何剩余的RTP包的合成时间等于当前RTP包和最早的RTP包的RTP时间戳差，时钟漂移修正类似于H.3.6.3。合成时间是指轨道的媒体时间线。

### H.3.6 Recording operations to facilitate inter-stream synchronization in playback

#### H.3.6.1 General

Lip同步，即在录制的RTP流之间的正确同步，至少可以通过以下两种方式实现：

1. 为每个RTP接收hint track生成一个RTCP接收hint track。在解析文件和解码和播放文件中包含的媒体流时，纠正不同流的RTP时戳时流之间的潜在时钟漂移。时钟漂移校正与同时接收和播放的RTP流的操作类似。这种操作模式对于记录单元来说很简单。然而，从一个精确的回放位置访问一个文件可能会更麻烦，因为它需要在访问时补偿所有记录的流的时钟漂移。
2. 通过修改一个或多个记录流的RTP时间戳，可以修正记录的RTP流之间的潜在时钟漂移。这种操作模式需要在录制时处理RTCP发件人报告，因此对于录音单元来说比创建RTCP接收hint track更乏味。然而，播放者的操作是很简单的。

记录单元应使用timestamp synchrony box[9.4.1.2]来指示已使用了哪种Lip同步方法。timestamp synchrony box包含timestamp\_sync字段。timestamp\_sync等于1表示播放者应该使用RTCP接收hint track进行Lip同步。timestamp\_sync等于2表示播放者应该使用合成时间戳来进行Lip同步。

一些实现可能会在实时记录操作期间首先创建RTCP接收hint track，然后通过修改RTP时间戳作为离线后处理步骤来补偿时钟漂移。

下面的子句提供了关于这两种方法的更多细节。

#### H.3.6.2 Facilitating lip synchronization based on RTCP Sender Reports

记录单元将特定RTP流的所有RTCP发送者报告作为相应RTCP接收hint track的样本。

#### H.3.6.3 Compensating clock drift in timestamps

不建议修改已录制的音频流的RTP时间戳。这样的修改会导致播放器中的音频时间尺度的修改，这是一个很重要的操作。

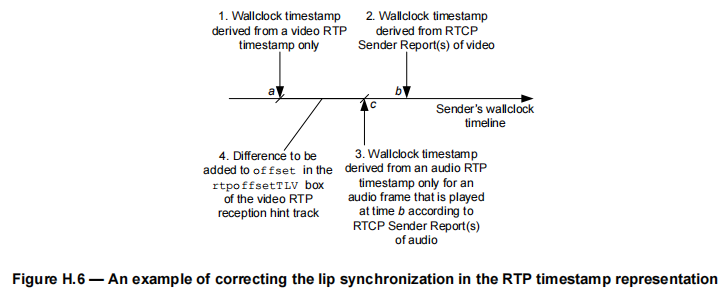
视频和其他非音频流的RTP时间戳的记录表示应使用以下程序进行修改。

首先，视频帧的挂钟时间戳a来自与视频帧对应的RTP时间戳，作为先前视频帧的挂钟时间戳和以挂钟时间轴为单位的当前和先前视频帧的RTP时间戳的差异。

第二，基于RTCP发送者报告导出视频帧在挂钟时间上的回放时间b。如果没有准确指示视频帧的挂钟时间的RTCP发送者报告可用，则可以外推挂钟时间，假设RTP时间戳时钟和RTCP发送者挂钟偏离的速率保持不变。

第三，基于音频的RTCP发件人报告，导出在挂钟时间轴的时间b与视频帧同时播放的音频RTP时间戳。不需要有一个完全具有派生的音频RTP时间戳的音频帧。音频样本的挂钟时间戳c从导出的音频RTP时间戳作为上述音频帧的挂钟时间戳和导出的音频RTP时间戳与前面音频帧的RTP时间戳的差值的总和计算。。

a和c之间的差异，如果有的话，应该在表示文件中的视频RTP时间戳的字段中进行补偿。在实践中，最简单的方法可能是将差异添加到rtpoffsetTLV box中的偏移字段中，如图H.6所示。另一种选项，重写Decoding Time to Sample box和Track Fragment Run boxes（如果有的话），实现起来可能更麻烦，因为通过样本计数和持续时间的组合来编码样本时间的方式很特殊，而且可能也需要更多的存储空间。



### H.3.7 Representation of reception times

如9.4.1.4所述，包的接收时间由包含包的样本的解码时间和包的RTP包结构的relative\_time值的总和表示。

最早接收到的RTP包的接收时间应为零，所有后续包的接收时间应相对于最早接收到的RTP包的接收时间。

接收时间的时钟源未定义，例如接收器的挂钟。如果一个接收hint track的接收时间范围与另一个接收hint track的接收时间范围完全或部分重叠，则这些hint track的时钟源应相同。

报文的接收时间应该对应于RTP(通常是UDP)下面的协议栈层输出报文的瞬间。

### H.3.8 Creation of media samples

根据相关RTP有效载荷规范和RTP本身的指示，从接收到的RTP数据包中创建媒体样本。然而，大多数媒体编码标准只指定了无错误流的解码，因此应该确保媒体track中的内容可以被任何符合标准的媒体解码器正确解码。因此，传输错误的处理需要两个步骤：检测传输错误和推断可以正确解码的样本。这些步骤将在后面的段落中进行描述。

可以从RTP序列号数值的间隙中检测到丢失的RTP数据包。包含比特错误的RTP数据包通常不会被转发到应用程序，因为它们的UDP校验和失败，并且数据包在接收器的协议栈中被丢弃。因此，在接收机中通常被视为误错误的包丢失。

可以正确解码的媒体样本的推断取决于媒体编码格式，因此在这里没有详细描述。通常，样本间预测在音频编码格式中较弱或不存在，而大多数视频编码格式则大量利用样本间预测。因此，在许多音频格式中丢失的样本通常可以被一个沉默或错误隐藏的音频样本所取代。应该分析视频包的丢失是否与非参考图片或参考图片有关，或者更一般地说，丢失发生在时间可扩展性层次中。然后应该得出结论，哪些图片可能不能正确解码。例如，非参考图片的丢失不会影响任何其他图片的解码，而在基准时间水平上的参考图片的丢失通常会影响所有图片，直到下一个可供随机访问的图片，例如H.264/AVC中的IDR图片。视频track不能包含任何依赖于任何丢失的视频样本的样本。

### H.3.9 Creation of hint samples referring to media samples

如H.3.8中所述，从接收到的RTP数据包中创建媒体样本。RTP接收hint track如H.3.4所述创建，但RTPpuck结构的内容取决于相应媒体样本的存在，如下所示。

如果接收到的RTP包的包有效负载在媒体track中表示，则相关包构造器的跟踪引用被设置为指向媒体track并通过引用包括包有效负载。为了节省存储空间并使文件编辑操作更容易实现，不建议在接收的RTP示例的extradata部分中包含包有效负载的副本。

如果接收到的RTP包的包有效载荷没有在媒体track中表示，则按照H.3.4中解释的方式创建RTP包结构的实例。

## H.4 Playing of recorded RTP streams

### H.4.1 Introduction

本条款描述了回放包含已记录的RTP流的文件所需的操作。其组织结构如下：

* 在播放RTP流之前，应该先分析文件的内容。特别是，应该识别代表同一媒体流的替代track，并选择其中一个track进行解码和回放。应该检测编码格式，以便预先知道它可以被播放者解码。这些准备操作在H.4.2中有更详细的描述。
* 如果正在处理RTP接收hint track，则如H.4.3所述。例如，应该适当地检测和处理数据包丢失。
* 应按照H.4.4的要求正确处理。
* 如果存储在文件中的RTP流是从流的开始以外的位置被访问的，则需要适当的流间同步和解码器初始化，如H.4.5所述。

### H.4.2 Preparation for the playback

在回放的准备阶段，播放者会选择所播放的track。首先解析该文件的基本track结构。这些track是根据它们所属的交替组进行分组的。属于同一备用组的track在track header box中用相同的alternate\_group值表示。从每个备用组中的一个track被选择来播放如下。

如果备选组中存在RTP接收hint track，则优选回放，因为它包含接收到的RTP流的整个表示，与来自接收到的RTP流的媒体跟踪不同，它可能使用接收到的RTP包的子集，可以由任何标准兼容解码器解码，而没有处理包丢失的能力。

应确保播放者与所选track的兼容性。例如，应该检查编解码器、配置文件和在track中使用的级别是否使播放者能够支持。

在RTP接收hint track中用于编码位流的编解码器、配置文件和级别可以从RTP流的SDP描述中得出结论。SDP描述存储在movie级的索引track中。如果SDP在整个文件中保持不变，则可以将其另外作为movie SDP信息和track SDP信息额外存储在用户数据框中。如果存在跟踪SDP信息，则可以对其进行解析，以查找RTP接收hint track中包含的位流所使用的编解码器、配置文件和级别。如果不存在movie SDP信息或跟踪SDP信息，则遍历移动级别索引跟踪以查找和解析每个SDP索引，从而解析用于RTP接收提示跟踪中包含的位流的编解码器、配置文件和级别。

如果替代组中不存在RTP接收hint track，则应该检查备用组中媒体track的样本条目或样本条目，以找出播放者能够支持哪些。

### H.4.3 Decoding of a sample within an RTP reception hint track

原始的RTP包可以从RTP接收提示样例中重建，方法是根据RTP包结构创建RTP包头，并解析RTP包结构的构造函数。 因此，文件播放器处理RTP接收hint track的一种方法是重新创建已接收的包流，并将重新创建的包流当作新接收的包流来处理。

RTP结构中包含的relative\_time字段可以用来调度RTP接收端将数据包插入缓冲区的时间。 然而,可能更可取的做法是修改记录RTP流的解码过程，这是使解码器输出缓冲区保持尽可能满的一种方式，为了避免由于延迟的数据包或实时解码中偶尔出现的问题而造成的中断或不稳定的播放，在运行其他进程的系统中除了播放者。

应从RTP序列号中的间隙中检测到数据包丢失。对数据包丢失的反应取决于特定的媒体解码器的实现，并且也可能取决于用户的偏好。

### H.4.4 Lip synchronization

要实现流之间的正确同步，需要执行以下步骤：

在播放开始时的音轨间同步。

一个track的媒体时间轴的起始位置可以在该文件的movie时间轴中移动，如以下两段所述。

对于媒体跟踪和RTP接收提示跟踪，如果没有与RTCP接收提示跟踪相关联，则应使用编辑列表框在移动时间轴中移动媒体时间轴的起始位置，如H.3.2所述。 通过解析曲目的Edit List box(如果存在的话)，选择用于播放的track的媒体时间线将映射到movie时间线。每个媒体track的回放以及每个RTP接收hint track，这个track 与RTCP接收提示track没有关联，从track的Edit List box中指定的movie时间轴位置开始或者从movie时间轴的开始，如果不存在该track的Edit List box。

对于与各自的RTCP接收hint track相关联的RTP接收hint track，媒体时间轴在movie时间轴内的起始位置的移动推断如下: 所有RTP接收hint track中包含最早RTP包的RTP接收hint track的媒体时间线在movie时间线内(即从movie时间线上的时间0开始)没有移动。 任何其他RTP接收hint track的媒体时间线起始时间等于当前track中最早RTP包与所有RTP接收hint track中最早RTP包的track的时间戳差。

重建媒体时间轴上重建RTP时间戳和合成时间(H.3.5)。

如果使用RTCP接收hint track，则基于RTCP发件人报告校正RTP时间戳和合成时间。

修正与H.3.6.3中描述的类似。但是，不是将时间a和c之间的差异添加到文件中RTP时间戳的表示中，而是在回放期间添加到movie时间轴上视频帧的表示时间中。

对解码的媒体样本的输出进行间隔。

建议以播放者挂钟的节奏播放录制好的节目，并使用音频播放时钟作为播放器的挂钟。 所述音频回放设置为在所述音频信号的本机采样频率处连续。 播放者的呈现时钟以音频播放的速度运行，也就是说，它的值总是等于(播放最频繁的未压缩音频样本的数量)×(音频信号的采样频率)。  视频track(和潜在的其他连续媒体track)的回放同步到播放者的显示时钟。 换句话说，当播放者的演示时钟在movie时间轴上满足视频样本的合成时间时，视频样本就会被播放出来。

只有一个文件同时被记录和回放,以及如果接收者wallclocks跑得比发送方wallclock快，可能不建议根据接收端墙上时钟的速率对播放进行节奏调整，可以按照以下步骤将接收端墙上时钟的速率同步到发送端墙上时钟的速率。

通过在接收时间(根据接收方时钟)和发送方各自的墙上时钟时间戳之间创建一个关系来恢复发送方时钟的速度，这些时间戳是由RTCP发送方报告(RTCP sender Reports)重建的。 建议使用音频播放时钟作为接收时钟。 由于网络和接收方的延迟可能是变化的，接收时间和发送方各自的时间戳之间的关系需要在大量接收的数据包上取平均值。 将接收时间和发送者各自的时间戳之间的关系求平均，得出时间刻度乘因子。

为每个样本导出接收器时钟的时间轴上的呈现时间。 如果使用了RTCP接收hint track，那么呈现时间就是影片时间轴上样本的合成时间，还包括上面步骤3中描述的时钟漂移校正。 如果没有使用RTCP接收hint track，则演示时间直接是影片时间轴上示例的合成时间。 然后，仅为播放目的，所有正在播放的曲目中样本的呈现时间应乘以时间刻度乘因子。

信号的时间拉伸应相应地进行。示例在其演示时间被播放。

在实践中，时间刻度乘因子和从RTP时间轴到发送方的挂钟的映射(上面的步骤3)可以作为一个单独的操作来实现。

### H.4.5 Random access

随机访问是指对文件中表示的媒体流的非线性访问。 换句话说，在一个随机访问操作中，文件是从另一个示例访问的，而不是以前播放的，或者文件最初是从一个不是movie时间轴开始的位置访问的。

建议向用户提供与文件的movie时间轴相关的随机访问功能，而不是任何其他时间轴，如发送方挂钟时间轴。 通过使用movie时间轴作为基础，随机访问操作的步骤数保持在较低的水平。

首先，它被推导出哪些媒体帧在一个期望的随机访问位置（或最接近它，如果没有确切地在一个期望的随机访问位置）。在媒体track、音频的RTP接收hint track以及任何timestamp\_sync字段等于2的RTP接收hint track的情况下，可以根据由Edit List box中指示的初始起始位置移动的合成时间戳（在媒体时间轴上）直接导出最接近所需随机访问位置的媒体帧。在timestamp\_sync字段等于1的非音频RTP接收hint track(指示使用RTCP接收hint track)的情况下，应按照H.4.4所述获得样本的呈现时间，直到找到最接近期望的随机访问位置的呈现时间。

第二，许多媒体位流的解码只能从特定类型的帧开始，例如H.264/AVC的IDR图像。因此，播放者实现可能有不同的方法，包括以下方法：

1. 在期望的随机接入位置或之前发现最近的帧，从该帧开始解码，并仅从期望的随机接入点开始渲染。这种方法可能意味着在开始渲染之前会有一些处理延迟。
2. 在所需的随机接入点或之后，使用可以开始解码的最早帧开始解码和渲染。通常，音频播放将比视频播放更早开始，但在渲染开始之前的处理延迟比前一个选项中的要小。

## H.5 Re-sending recorded RTP streams

### H.5.1 Introduction

将之前记录过的RTP流重新发送到文件中可能是一个理想的操作。例如，如果RTP流通过广播或流媒体服务接收并记录到文件中，则可能希望使用WLAN连接将它们从一个设备重新发送到家庭环境中的另一个设备。本条款提供了重新发送已记录的RTP流的建议。

一种基于RTP的通信系统，包括一个源端点（也称发送者）和目标端点（也就是一个接收器），也可以包含一个或多个混合器和翻译器。发送方和接收方是RTP和RTCP会话的端点。RTP翻译器和混合器的行为在RFC3550中指定，并在RFC5117中澄清。一般来说，接收RTP流并将其存储到文件中的记录单元作为目标端点，并且重新发送单元从文件中读取存储的RTP流并发送它们作为源。通常，重新发送的RTP流的有效负载不被修改，这使得记录单元和重新发送单元的组合具有类似于RFC5117中所述的传输转换器的作用。然而，翻译器的基本特征是接收器不能检测到它的存在。因此，记录单元和重新发送单元的组合不能作为传输转换器，除非重新发送与原始流的记录同时发生。由于这种情况非常罕见，本条款中的讨论将记录单元视为终止原始RT的目的地。

本条款的组织结构如下：

* H.5.2包括如何从RTP接收hint track组成RTP包以及如何调度RTP包的传输的建议。
* H.5.3讨论了如何生成RTCP包以及如何处理接收到的RTCP包。

### H.5.2 Re-sending RTP packets

建议按如下方式构造和传输这些数据包。

建议根据存储在接收hint track中的构造函数构造包载荷，即，建议包载荷与接收的包相同，除非不同的包大小对重新发送包的网络至关重要。

* RTP报文的报头字段的值由RTP接收hint track建议，应该保持与相应的RTP报文结构相同，除了以下情况:
* 最初的RTP时间戳偏移和RTP序列号偏移应选择随机的值，无论RTP接收提示的任何数据包的RTPpacket结构的RTPsequenceseed字段的值存储在参考的接收hint示例条目的'tsro' box的偏移字段的值或RTPsequenceseed字段的值。
* RTP时间戳字段的值应该是随机初始偏移量、RTP包结构中的偏移量和各自RTP样本的解码时间的和。 如果和超过32位无符号整数的最大值，则应该对其进行换行。
* RTP序列号的相对增量应该与RTPsequenceseed字段中记录的值相同。 因此，如果在记录的流中有丢包，被重发的流在RTP序列号中也有相应的间隙，并且接收端能够推断出丢包。
* CSRC计数字段的值应该始终为零，因为之前记录的RTP会话的贡献源没有积极地修改被重新发送的RTP会话的流。 源标识符空间(对于SSRC和CSRC)是特定于会话的。 因此，RTP报头的CSRC列表应该为空，而不管接收流可能存储的CSRC值，这些值包含在rtpackets结构中的received CSRC TLV box中。
* 载荷类型字段的值可以根据所使用的信令方案动态选择。
* SSRC字段的值应该是随机选择的，潜在的冲突应该按照RFC 3550的规定处理。 接收到的流的SSRC值可以存储在接收hint示例条目的ReceivedSsrcBox中，但是当流被重新发送时，它应该被忽略。
* 记录的RTP报头扩展，存储在rtppack结构中的rtphdrextTLV中，如果有的话，只有当重新发送单元能够验证它们对重新发送的流是有效的时，才应该重新发送。 如果重新发送单元不能解析记录的RTP报头扩展的语义，它们不应该被重新发送。

数据包的接收时间,由解码时间之和的RTP接待hint示例包含包和relative\_time RTPpacket结构的值,等于数据包的传输时间和误差造成的传输延迟和处理延迟协议栈的接收器。 由于传输延迟抖动和处理延迟的变化，相邻数据包的误差可能不相等。此外，接收流时使用的协议栈可能与重新发送流时使用的协议栈不同。 由于这些原因，接收时间通常不适用于重新发送数据包的传输速度（pace the transmission）。 在所有情况下，重发单元应该验证重发的数据包流是否符合正在使用的缓冲模型，如果有的话。如果重发单元可以断定接收流时使用的网络环境和协议栈与重发送记录流时使用的网络环境和协议栈相似，则接收时间可作为调度包传输的基础。 重发单元应努力消除或隐藏所记录的流中的传输延迟抖动。 如果重发单元无法断定接收流和重发记录流时使用的网络环境和协议栈类似或不确定哪种包调度合适，则可以使用解码时间作为调度的基础。

### H.5.3 RTCP Processing

RTCP发件人报告和其他RTCP消息是按照RFC3550中指定的约束重新生成的，而不是直接使用在RTCP接收hint track中记录的RTCP消息，如果有的话。

RTCP发件人报告包含发送报告时的挂钟时间和与指示挂钟时间相同的时间对应的RTP时间戳。生成的RTCP发件人报告的RTP时间戳，如下所示。为RTCP发送方报告中指示的挂钟时间对应的样本提供参考时钟时间线上的表示时间。参考时钟可以是在会话开始时初始化为0的重新发送单元的挂钟。对应的提示时间的样本可能不在对应的RTP接收hint track中，因为RTP接收hint track中对应的样本的采样实例可能与RTCP发件人报告的传输实例不匹配。然而，根据RFC3550的指示，RTP时间戳的导出就好像在RTP流中有一个样本对应于指示的挂钟时间。RTCP发件人报告的RTP时间戳应该从RTCP发件人报告中指示的挂钟时间前后的样本的RTP时间戳中进行线性插值。为了在样品之前和之后的结论

在处理接收到的RTCP接收器报告时，应该注意到，报告的累计丢失的数据包数还包括最初从未接收到的未发送的数据包，并对应于RTP接收提示轨道中的RTP序列号中的间隙。任何拥塞管理、重传或其他数据包丢失恢复方法都应该考虑到这一点。

# Annex I

(normative)

**Stream Access Points**

## I.1 Introduction

本附件定义了流接入点(SAP)，并指定了六种类型的SAP。

流访问点(SAP)允许对媒体流容器的随机访问。容器可以包含多个媒体流，每个媒体流都是具有特定媒体类型的连续媒体的编码版本。SAP是容器中的一个位置，允许仅使用(a)从该位置开始包含的信息，以及(b)可能的初始化数据，或外部可用的初始化数据。派生的规范应该指定是否需要初始化数据来访问SAP上的容器，以及如何访问初始化数据。

## I.2 SAP properties

对于每个SAP的属性，ISAP, TSAP, ISAU, TDEC, TEPT 与TPTF 是否被标识并定义为：

* TSAP是媒体流的任何访问单元的最早呈现时间，这样呈现时间大于或等于TSAP的媒体流的所有访问单元都可以使用从ISAP开始的位流中的数据正确解码。
* ISAP是比特流中最大的位置，因此，表示时间大于或等于TSAP的媒体流的所有访问单元都可以使用从ISAP开始的比特流数据，而在ISAP之前没有数据来正确解码。
* ISAU是在媒体流中按解码顺序的最新访问单元的比特流中的起始位置，使得可以使用呈现时间大于或等于TSAP的媒体流的所有访问单元正确解码的最新的访问单元。

注： ISAU始终大于或等于ISAP。

* TDEC是媒体流的任何访问单元的最早呈现时间，可以使用比特流中的数据正确解码。
* TEPT是比特流中从 ISAU开始的媒体流的任何访问单元的最早表示时间。
* TPTF是从 ISAU开始以解码顺序进行的媒体流的第一访问单元的呈现时间。

## I.3 SAP types

六种类型的sap被定义，它们的属性如下：

Type 1: TEPT = TDEC = TSAP = TPTF

Type 2: TEPT = TDEC = TSAP < TPTF

Type 3: TEPT < TDEC = TSAP <= TPTF

Type 4: TEPT <= TPTF < TDEC = TSAP

Type 5: TEPT = TDEC < TSAP

Type 6: TEPT < TDEC < TSAP

注意：SAP的类型仅取决于访问单元可正确解码以及按显示顺序排列。这些类型非正式地对应于一些常见的术语：

Type 1对应于一些编码方案中所知的“封闭GoP随机接入点”(其中所有访问单元，按解码顺序，从ISAP开始可以正确解码，导致正确解码的访问单元没有间隙的连续时间序列)，此外，解码顺序的访问单元也是表示顺序的第一访问单元。

Type 2对应于在某些编码方案中所知道的“封闭的GoP随机接入点”，其中从 ISAU开始的媒体流中按解码顺序的第一接入单元不是按呈现顺序的第一接入单元。

Type 3对应于一些编码方案中所谓的“开放GoP随机接入点”，其中一些在ISAU之后的访问单元不能正确解码，呈现时间小于TSAP。

Type 4对应于一些编码方案中所谓的“渐进解码刷新(GDR)随机接入点”，其中有一些从ISAU开始和之后的解码顺序的访问单元不能正确解码，呈现时间小于TSAP。

Type 5对应于从ISAP开始的解码顺序不能正确解码且呈现时间大于TDEC的情况，TDEC是从 ISAU开始的任何访问单元的最早呈现时间。

Type 6对应于以下情况：从ISAP开始的解码顺序不能正确解码，呈现时间大于TDEC，并且TDEC不是从ISAU开始的任何访问单元的最早呈现时间。

# Annex J

(normative)

**MIME Type Registration of Segments**

## J.1 Introduction

本附件提供了根据8.16格式化的媒体段的正式MIME注册。

## J.2 Registration

MIME media type name: video

MIME subtype name: iso.segment

Required parameters: none

Optional parameters: as specified by RFC 6381 and its successors

Encoding considerations: as for video/mp4

Security considerations: See section 5 of RFC 4337.

Interoperability considerations: A number of interoperating implementations exist

within the ISO/IEC 14496 community, and that community has reference

software for reading and writing the file format.

Published specification: ISO/IEC 14496-12:2012 (expected)

Applications: Multimedia

Additional information:

Magic number(s): none

File extension(s): m4s

Macintosh File Type Code(s): None

Person to contact for info: David Singer, singer@apple.com

Intended usage: Common

Author/Change controller: David Singer, ISO/IEC 14496 file format chair

## 参考文档

[1] The QuickTime file format specification, in PDF:

<http://developer.apple.com/documentation/QuickTime/QTFF/qtff.pdf>

[2] 3GPP TS 26.244, 3GPP file format (3GP)

[3] 3GPP TS 26.346, Multimedia Broadcast/Multicast Service (MBMS); Protocols and codecs

[4] OMA BCAST\_Distribution-V1\_0: File and Stream Distribution for Mobile Broadcast Services

[5] IETF RFC 3926, FLUTE - File Delivery over Unidirectional Transport, October 2004

[6] IETF RFC 3450, Asynchronous Layered Coding (ALC) Protocol Instantiation, December 2002

[7] IETF RFC 3451, Layered Coding Transport (LCT) Building Block, December 2002

[8] IETF RFC 3452, Forward Error Correction (FEC) Building Block, December 2002

[9] IETF RFC 3695, Compact Forward Error Correction (FEC) Schemes, February 2004

[10] IETF RFC 1864, The Content-MD5 Header Field, October 1995

[11] IETF RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1, June 1999

[12] IETF RFC 3061, A URN Namespace of Object Identifiers, February 2001

[13] IETF RFC 3550, RTP: A Transport Protocol for Real-Time Applications, July 2003

[14] IETF RFC 3551, RTP Profile for Audio and Video Conferences with Minimal Control, July 2003

[15] IETF RFC 4122, A Universally Unique IDentifier (UUID) URN Namespace, July 2005

[16] IETF RFC 4771, Integrity Transform Carrying Roll-Over Counter for the Secure Real-time Transport

Protocol (SRTP), January 2007

[17] IETF RFC 5119, A Uniform Resource Name (URN) Namespace for the Society of Motion Picture and

Television Engineers (SMPTE), February 2008

[18] ICC.1:2001-04, File format for color profiles, International Color Consortium

[19] SMPTE RP 177, Derivation of Basic Television Color Equations; Society of Motion Picture and

Television Engineers (SMPTE), 1993

[20] ISO/IEC 13818-1, *Information technology — Generic coding of moving pictures and associated audio*

*information — Systems*

[21] ISO/IEC 14496-15, *Information technology — Coding of audio-visual objects — Advanced Video*

*Coding (AVC) file format*

[22] IETF RFC 5117, *RTP Topologies*, WESTERLUND, M. et al., January 2008.